Spectrum Sharing and SAS-CBSD Interface Simulation

Hao Huang

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.
Espoo 18.9.2016

Thesis supervisor:

Prof. Risto Wichman

Thesis advisor:

M.Sc. (Tech.) Jaakko Ojaniemi

Aalto University
School of Electrical
Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Hao Huang

Title: Spectrum Sharing and SAS-CBSD Interface Simulation

Date: 18.9.2016 Language: English Number of pages: 8+86

Department of Signal Processing and Acoustics

Professorship: Signal Processing

Supervisor: Prof. Risto Wichman

Advisor: M.Sc. (Tech.) Jaakko Ojaniemi

Spectrum Sharing technologies enables more dynamic spectrum management regulation and framework to provide capacity for the ever-increasing demand of mobile data traffic. This thesis reviewed the background and current state of the development of Spectrum Sharing approaches. TVWS, LSA and CBRS were examined in detail as the most representative solutions.

The thesis compared architectural similarities and differences between LSA and CBRS. The thesis reviewed SAS-CBSD interface protocol and continued with a practical validation of SAS-CBSD interface specification. By implementing the interface in Python, interface simulation was conducted via the assistance of automation scripts. The thesis concluded that the SAS-CBSD interface is functioning as designed, noting that ESC will further extend the spectrum access dynamism. The thesis also pointed out the need to specify SAS-relevant data models for database API standardization.

Keywords: Spectrum Sharing, TVWS, LSA, CBRS, SAS, Python

Preface

For my mom.

Otaniemi, 18.09.2016

Huang, Hao

Contents

Abstract ii

Preface iv

Contents v

Symbols and abbreviations vii

1 Introduction 1

2 Spectrum Sharing Overview 3

2.1 Dynamic Spectrum Access Radio . . . 3

2.1.1 Software Defined Radio . . . 3

2.1.2 Cognitive Radio System . . . 4

2.2 Spectrum Shortage and Resolutions . . . 5

2.3 White Space in UHF band . . . 10

2.3.1 White Space Device . . . 10

2.3.2 TVWS Access Methods . . . 11

2.4 Spectrum Sharing Approaches in Practice . . . 13

2.4.1 TVWS in UK . . . 14

2.4.2 TVWS in US . . . 16

2.4.3 Licensed Shared Access . . . 17

2.4.4 Citizens Broadband Radio Service . . . 19

2.4.5 Other Spectrum Sharing Approaches . . . 20

2.5 Summary . . . 23

3 Architectural Comparison of LSA and CBRS 25

3.1 Spectrum Sharing Framework . . . 25

3.2 System Design . . . 26

3.3 Incumbent Protection . . . 29

3.4 Use Cases . . . 31

4 SAS-CBSD Protocol 33

4.1 CBRS Functional Architecture . . . 33

4.2 SAS-CBSD Interface Procedures . . . 34

4.2.1 Pre-requisite Procedures . . . 34

4.2.2 SAS Discovery and CBSD Registration . . . 35

4.2.3 CBSD Spectrum Request and Relinquish . . . 35
4.2.4 SAS Spectrum Reassignment/Revocation . . . 36
4.3 CBSD State and State Transition . . . 36

5 SAS-CBSD Interface Simulation 39
5.1 Simulation Setup . . . 39
5.1.1 Robot Framework Installation . . . 39
5.1.2 Keyword-Driven Automation . . . 40
5.1.3 Automation Script Layout . . . 40
5.2 Simulation Core Modules . . . 42
5.2.1 SpectrumAccessSystem . . . 42
5.2.2 SASLocationService . . . 43
5.2.3 SASDatabase . . . 43
5.2.4 CBSD Meta Data . . . 44
5.3 Simulation Implementation . . . 44
5.3.1 CBSD State Transition . . . 45
5.3.2 Geolocation . . . 51
5.3.3 Incumbent Protection . . . 55
5.4 Simulated Scenario and Results . . . 56
5.4.1 CBRS First Phase Simulation . . . 56
5.4.2 CBRS Second Phase Simulation . . . 60
5.4.3 Conclusion . . . 63

6 Summary 65

References 67

A Appendix 1: SAS-CBSD Simulation Script 72

B Appendix 2: SpectrumAccessSystem.py 73

C Appendix 3: SASLocationService.py 78

D Appendix 4: SASDatabase.py 83

Symbols and abbreviations

Abbreviations

3GPP	3rd Generation Partnership Project
API	Application Programming Interface
CBSD	Citizens Broadband Radio Service Device
CBRS	Citizen's Broadband Radio Service
CEPT	European Conference of Postal and Telecommunications Administrations
CRS	Cognitive Radio System
DTT	Digital Terrestrial Television
ECC	Electronic Communications Committee
ESC	Environmental Sensing Capability
ETSI	European Telecommunications Standards Institute
FCC	Federal Communications Commission
FDD	Frequency Division Duplex
GAA	General Authorized Access
GIS	Geographic Information System
HSPA	High Speed Packet Access
IDE	Integrated Development Environment
IMT	International Mobile Telecommunications
IoT	Internet of Things
ITU-R	International Telecommunication Union Radiocommunication Sector
JAR	Java Archive
JSON	JavaScript Object Notation
LSA	Licensed Shared Access
LTE	Long-Term Evolution
M2M	Machine To Machine
MBMS	Multimedia Broadcast Multicast Service
MIMO	Multiple-Input and Multiple-Output
MNO	Mobile Network Operator
MVNO	Mobile Virtual Network Operator
HetNet	Heterogeneous Networks
NRA	National Regulatory Authority
NTIA	National Telecommunications and Information Administration
Ofcom	The Office of Communications
OFDM	Orthogonal frequency-division multiplexing
OTT	Over-The-Top

PA	Priority Access
PCAST	President's Council of Advisors on Science and Technology
PDSCH	Physical Downlink Shared Channel
PMCH	Physical Multicast Channel
QoS	Quality of Service
RSPG	Radio Spectrum Policy Group
SAS	Spectrum Access System
SC-FDMA	Single-carrier Frequency Division Multiple Access
SDR	Software-Defined Radio
SON	Self Organized Networks
PAL	Priority Access License
PMSE	Programme Making and Special Events
TDD	Time Division Duplex
TVWS	TV White Space
UHF	Ultra High Frequency
VHF	Very High Frequency
WSD	White Space Device
WSDB	White Space Databases

1 Introduction

Wireless communication is becoming increasingly important in providing broadband coverage to end users. As the demand of mobile communication grow, spectrum available for mobile operators to expand network capacity is becoming increasingly scarce. Projection shows that a spectrum shortage will be induced if traditional auction and license spectrum allocation model remains the only option to assign radio resources. This reality calls for more flexible and dynamic spectrum management and usage frameworks. This thesis examined the reasoning of the projected spectrum shortage as well as the proposed resolution by implementing Spectrum Sharing technologies. The thesis focused on reviewing the background and current state of Spectrum Sharing approaches. The thesis provided architectural comparison of the most prominent Spectrum Sharing applications. In addition, this thesis provided a practical verification of, one of the Spectrum Sharing Access system applications, Citizens Broadband Radio Service (CBRS).

To address the need of introducing additional spectrum for mobile communication via dynamic sharing, spectrum sharing technolgy is first deployed in UHF band in TV White Space(TVWS). The rationale for TVWS Spectrum Sharing is to utilize the spectrum made available as a result of the transition from Analog TV broadcast to Digital as well as the superior propagation characteristics of the UHF bands. Trials were held around the world with primary case studies conducted in the UK and in the US. TVWS Spectrum Sharing is yet to witness commercial success due to various reasons. Meanwhile Licensed Shared Access(LSA) and CBRS have been introduced on LTE-compatible bands, primarily targeting small-cell application complementing primary mobile network. Both frameworks employs tiered shared access focusing on co-existence of incumbents and spectrum shared access users such as mobile operators. Other spectrum sharing approaches focusing on reusing existing technologies and frameworks have also been introduced and serves as supplemental extension of the primary mobile network.

LSA and SAS shares architectural similarities while having differences in their design. They are both aimed at making traditionally licensed spectrum available to mobile network, while avoiding the cost of refarming and re-allocation, by providing mechanisms and incentives for the co-existence of incumbent and shared access user. They are both built with geolocation as primary shared access methods. LSA is a two-tier access model complimented with long term license agreement based shared

access. LSA is largely reserved for mobile operator to share spectrum access with incumbents. CBRS is a three-tier access model, with the third tier is reserved for general unlicensed access. Compare with LSA, the approach applied in CBRS will increase the level of architectural and regulatory complexity while allowing more flexible use of the shared bands.

CBRS operates on 3.5-3.7 GHz band. The band is currently occupied by military and government users. As a geolocation enabled spectrum sharing framework, CBRS network consists of several functional components. The main network control unit is the Spectrum Access System(SAS). SAS maintains connection to central databases and keeps records of incumbent protection parameters. SAS provides access control and communication with shared access devices referred as CBSD. SAS is enabled by geolocation database containing data for access determination. SAS is optionally supported by Incumbent Detection module equiped with sensing technologies, listening for undisclosed incumbent users, such as a Naval radar platform operating at sea, to protect from unwanted interference.

CBRS functional architecture and SAS-CBSD interface specifications were validated via process simulation. The simulation demonstrated that the system can be operated based on specified interface message structure and state flow. It is observed from simulation that, on a geolocation basis, an ESC sensing network will extend the dynamic spectrum usage and incumbent protection of CBRS.

2 Spectrum Sharing Overview

The growth of mobile broadband service usage is generating significant traffic on mobile networks. This trend is expected to continue. Auctioned spectrum resource available at Mobile Network Operators(MNO) is in a time of scarcity. Continuing to solely relying on current spectrum management regulation framework will result in a deficit of spectrum availability[10]. As a result, enhanced spectrum policies that promote spectrum usage efficiency and grants access to additional spectrum bands is needed.

New spectrum management approaches based on shared-access have been introduced worldwide. Several spectrum sharing mechanisms were implemented in UHF as well as LTE bands. They share similarities such as incumbent co-existence framework, access based on geolocation and can be applied as complementary off-loading for primary mobile networks constituting an integral part of next generation radio access framework.

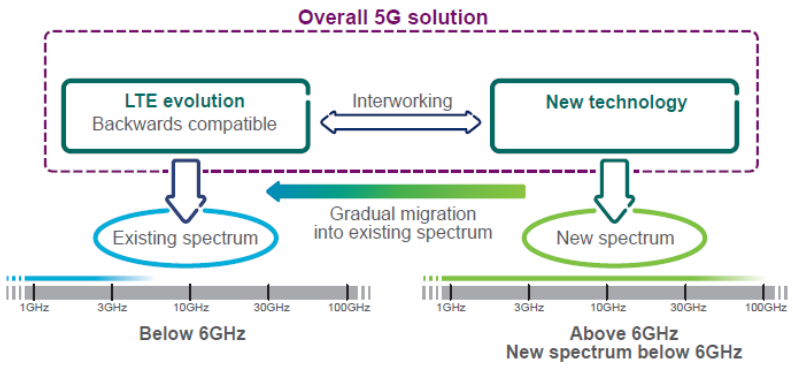

Figure 1: New spectrum access technology as part of 5G [3]

2.1 Dynamic Spectrum Access Radio

Radio communication technology is advancing towards more efficient and dynamic use of radio resources. The underling radio technology concepts enabling dynamic radio access are reviewed in this chapter.

2.1.1 Software Defined Radio

Software-defined radio (SDR) is a radio transmitter and/or receiver employing a technology that allows the RF operating parameters e.g. frequency range, modulation

type, or output power to be configured by software during operation [4]. The concept of Software Defined Radio is first introduced by Mitola in 1992. Enabled by the evolving digital electronic technologies, SDR provides a solution to accommodate increasing demands on radio equipment to support more bands, more spectrum and more waveforms.[5]

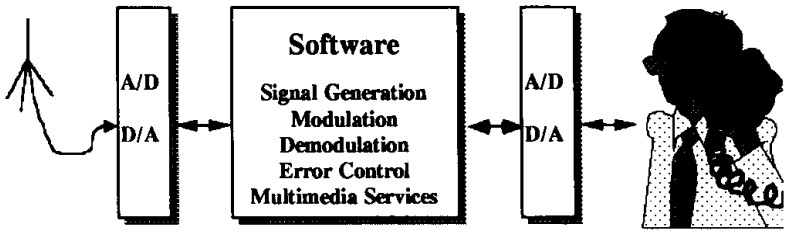

Figure 2: Software Defined Radio [6]

Most of the radios in the world today, including handsets, are based on SDR technology. Physical baseband processing is performed on a programmable Digital Signal Processor (DSP) to various degrees. The enabling technology behind – the Software Communications Architecture (SCA) enables SDR's reconfigurability. It functions as the middleware that enables radio waveforms used in communication to be defined in software. Beyond SDR, the challenge of introducing additional communication capacity in a finite amount of available spectrum remains to be addressed. Industry experts suggested that cognitive radio will contribute to the resolution.[7]

2.1.2 Cognitive Radio System

Cognitive Radio System (CRS) is a radio technology that allows the employing system to obtain knowledge of its operational and geographical environment, established policies and its internal state. CRS dynamically and autonomously adjust its operational parameters and protocols according to its knowledge in order to achieve predefined objectives and to learn from the results obtained[4].

Cognitive radio is considered as a goal towards which a software-defined radio platform should evolve: a fully reconfigurable wireless transceiver which automatically adapts its communication parameters to network and user demands. Originally designed as a software-defined radio extension, CRS adds dynamic spectrum access

ability to an SDR. Extensive researches have been done on dynamic radio access via sensing, focusing on spectrum-sensing cognitive radio (for example in the TV bands).

However it was observed that with cognitive radio access system relying purely on sensing is not sufficiently reliable due to local shadowing and other practical challenges. Spectrum sharing solutions currently deployed are all primarily geolocation based.

2.2 Spectrum Shortage and Resolutions

Mobile communications contributes to overall economic and social developments of both developed and developing countries. The evolution towards mobile broadband results in a new opportunity to bridge the gap between Internet-connected and unconnected people[17]. The telecommunication industry is being shaped by the growing demand of wireless data traffic and subsequent growing deficit of available spectrum. Technological standards, such as 4G LTE, are continuously evolving to provide faster connection speed and more efficient use of spectrum. Meanwhile, the demand for wireless data traffic is doubling annually. Cisco reported in 2016 that global mobile data traffic grew 74% in 2015 and over 4000 fold over the past ten years.[20]

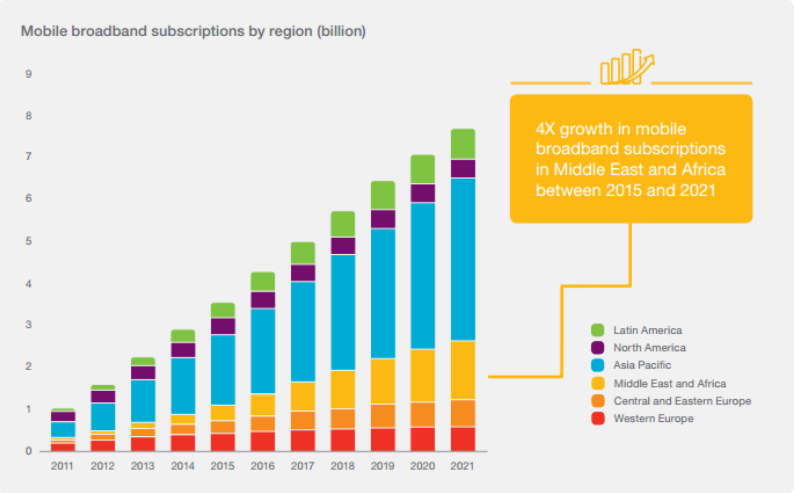

Figure 3: Mobile Broadband Subscription by Region [24]

ITU-R outlined main drivers of the increasing of mobile data traffic in IMT traffic estimation report. The growing demand will be driven by fast growing use base of mobile devices, increasing traffic of high-resolution video streaming and uptake

on mobile application download and usage[17]. This is evident in above Figure 3, illustrating mobile broadband subscription growth projection by region.

In addition to the main drivers, ITU-R report listed trends which will drive mobile data traffic increase and shape overall communication industry beyond 2020, illustrated in Figure 4. Machine to Machine applications and devices are one of the fastest growing segment of mobile data traffic usage. Machinery such as cranes, cars and home appliances will connect to internet and essentially are able to communicate to each other, under the broad term of Internet of Things. Cloud computing will also increase demand of mobile data, as the data storage and computation are provided via the internet (from the cloud) as services to user devices and terminals. Cloud computing requires always-on internet connection, regardless of time and location, as well as broad bandwidth to handle the volume of data to be exchanged. The trend of urbanization and related shifting demographics would also mean increased demand of mobile data traffic, as more people will become connected.

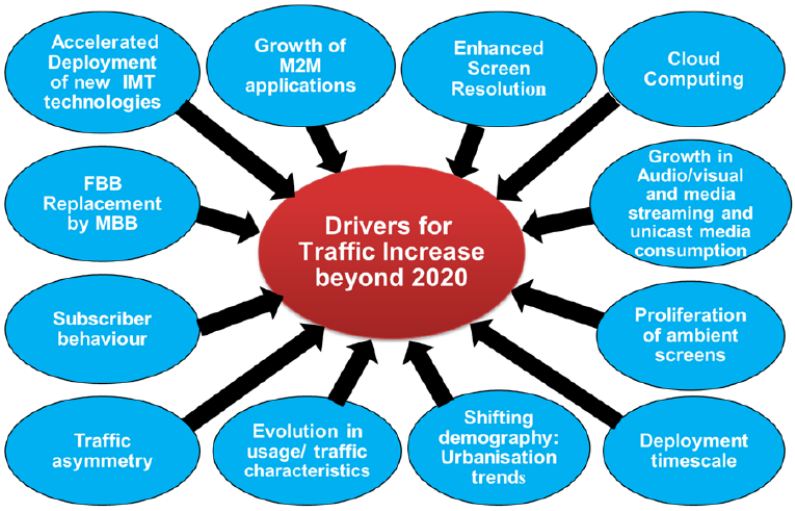

Figure 4: Drivers for Traffic Increase [17]

Sighting the growing demand of spectrum to carry mobile data traffic, Federal Communications Commission projected in 2010 a licensed spectrum deficit of almost 300 MHz in US by 2014.[19] Using the same FCC's formula and approach, forecast made in 2015 shows that by 2019, more than 350 additional MHz of licensed spectrum is needed in the US to support projected commercial mobile wireless demand.[10]

In 2010, an US presidential policy directive[12] was issued, requesting that ad-

ditional 500 MHz spectrum should be made available for both mobile and fixed wireless broadband use. The spectrum must be available to be licensed by the FCC for exclusive use or made available for shared access by commercial and government users. Both licensed or unlicensed wireless broadband technologies can be deployed. In 2012, White House PCAST spectrum report indicated that clearing and relocating Federal spectrum would not be sustainable due to incurred cost, time and disruption to Federal missions. The report signals evolving spectrum management approach away from the traditional licensed allocation model, illustrated in Figure 5. The report suggested that the essential elements of a new spectrum governance structure will be sharing and managed, over exclusivity and fragmentation.[18]

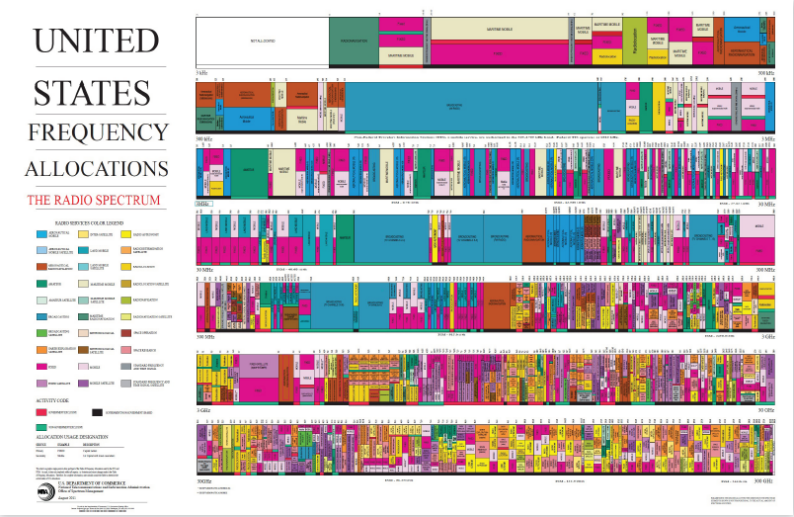

Figure 5: US Frenquency Allocation As-is [18]

PCAST report stated that two trends in communication technology advancement will enable reliable spectrum sharing architecture. Firstly, many wireless services are moving from cell towers covering large area to small cells as illustrated in Figure 6. As the services are provided to small geographical areas, the possibility of interference is reduced. Secondly, radio communication performance improvement made it possible to deliver service in the presence of signals from other systems. It is worth noting that NTIA report shows that if 3550-3650 MHz band is auctioned for traditional wireless operator's high-power transmission, an exclusion zone of 200 miles in land on US' East coast would be required to protect military incumbents. Therefore a small-cell, low power usage would be commercially more viable as it could significantly reduce the size of elimination zone.

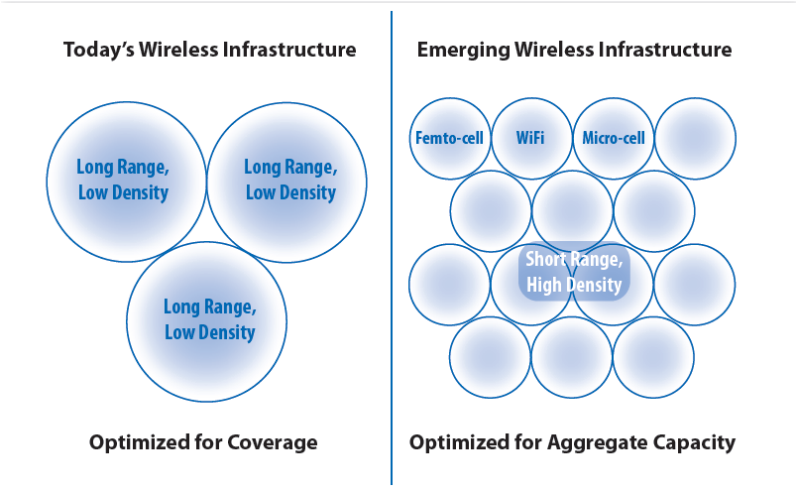

Figure 6: Small cell [18]

PCAST related its proposal to the shift from a collection of non-interconnected country road to a multi-lane highway of spectrum, where all types of radio services shares the same wide band. It argued that stating the problem is a shortage of spectrum is fundamentally incorrect[18]. Rather, the question is how spectrum can be better managed and more efficiently utilized. In order to set up the foundation of a high-speed shared spectrum access, wide bands of spectrum should be made available. The report recommended prioritizing creation of wideband in 2700 to 3700 MHz in the US by combining bands identified from NTIA FastTrack report[21]. This is illustrated in Figure 7

- 2700-2900 MHz: Aviation, aircraft landing and safety systems.[22]
- 2900-3300 MHz: Maritime, Radar
- 3300-3500 MHz: Radar, Amateur Radio, Fixed Satellite
- 3500-3650 MHz: Fixed Satellite, Radar

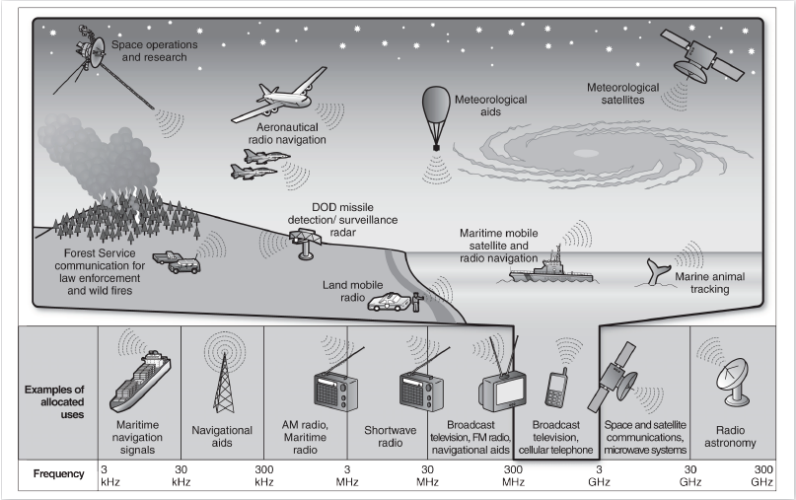

Figure 7: Shared Spectrum Wideband Identified[18]

When a Federal spectrum band is made available for high-speed shared access, Federal, military or national services (incumbents) will continue their operations within while other services shares the remaining of capacity available in the time and space domains. To govern shared spectrum usage, PCAST recommended a Spectrum Access System structure and a three-tier access model, illustrated in Figure 8. This would allow dynamic sharing of spectrum while ensuring incumbent services are prioritized and protected from interferences from other services and users on the same band.

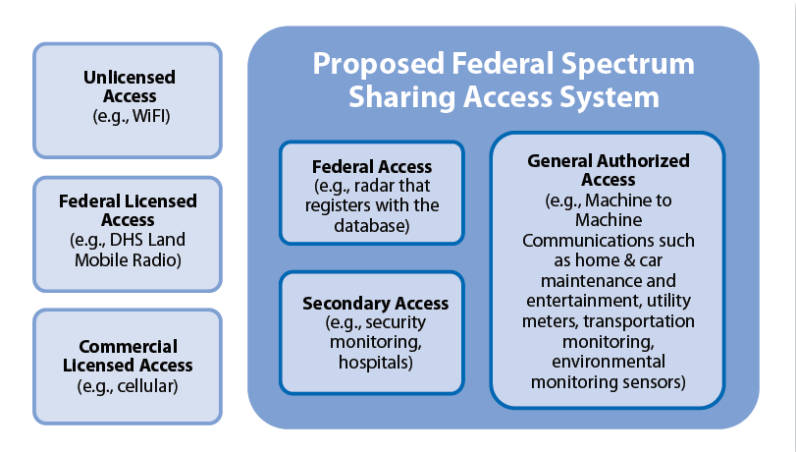

Figure 8: Proposed Spectrum Access System [18]

2.3 White Space in UHF band

Terrestrial broadcasting uses spectrum in the UHF band (470-862 MHz) and in the VHF band (173-230 MHz) for delivering analogue television signals. Television broadcasting transition to digital from analogue has finalized in most countries, with ITU GE06 agreement setting cut-off date in mid-2015.[43] Spectrum usage of digital terrestrial broadcasting is more efficient as one transmission channel of 6-8 MHz for one analogue program can carry a multiplex of up to 20 digital programs. The amount of spectrum made available from this transition is referred as the digital dividend.

TV White Spaces (TVWS) are UHF bands made available for radio communication service at given time and location where the usage would not cause interference to primary services, such as television broadcasting. TVWS spectrum provides more favourable propagation characteristics than shorter wavelengths, able to propagate further and go through walls. Aiming at maximizing the benefit of the digital dividend, both Radio Spectrum Policy Group(RSPG) and European Conference of Postal and Telecommunications Administrations(CEPT) provided their recommendations on TVWS operations.[8]

2.3.1 White Space Device

The fundamental uniqueness of Cognitive Radio technology is its ability to automatically adapt to a given spectrum and network environment. Cognitive radio system can be implemented collaboratively or non-collaboratively. In the collaborative approach, cognitive radio device and other relevant radio devices exchange mutual information regarding the frequency and time of usage of the spectrum. This approach requires a common protocol to negotiate these parameters. In the non-collaborative approach, a cognitive radio device senses the radio environment and determines by itself the radio parameters without communication with possible other users of the spectrum. [8]

Cognitive Radio operating in white space is referred as White Space Device (WSD). Evolved from the above principles, WSD cognitive techniques are employed to enable license exempt operation at TVWS[8] via both standalone sensing and centralized database assignment. ECC report 159[9] provided technical and operational requirements for cognitive radio systems in the white spaces of UHF band, ensuring the

protection of the digital broadcasting, PMSE services, Radio Astronomy in the 608-614 MHz, Aeronautical Radio navigation 645-790 MHz and Radio services in bands adjacent to the band 470-790 MHz.

- Spectrum sensing: if a device senses another transmission on the frequency it is using or on the adjacent frequencies, it automatically switches to another channel. The detection threshold has to account for sufficient margin to prevent interference, taking into account potential interference distances and propagation variations.
- Geo-location: devices aware of their locations by using a radio-navigation system such as GPS, can avoid transmitting on broadcast channels in their vicinity by referencing to a location-based spectrum utilization database.
- Local beacon: a local beacon transmitter operating in an unoccupied television channel broadcasts information to licence-exempt devices operating nearby.

Devices using cognitive techniques should be able to sense their environment and adjust their operating parameters (e.g. transmit power and frequency) to communicate with other devices on a non-interfering basis. This includes interference into terrestrial TV (analogue and digital) as well as interference to Programme making and special events(PMSE) services.[8]

Possible WSD deployment scenarios and various incumbent service protection methodologies were evaluated in ECC report 159 [9]. WSD usage can be categorized into Personal/Portable, Home/Office and Access point. Personal/Portable WSDs, much like smart phone or laptop, provides mobile data access to internet content browsing and certain machine to machine communication scenarios such as electronic payment. Home and office appliances such as TV and printer, can benefit from connectivity brought by their WSD capabilities. WSD can also act as access points providing "the last mile"[25] delivering wireless internet connectivity to the end users from e.g. high-speed fibre optic network.

2.3.2 TVWS Access Methods

Cognitive approaches which can be used by WSD to detect and access unoccupied spectrum were assessed in depth in ECC report. Three approaches, spectrum sensing, geolocation database and beacon, were reviewed in terms of operative and technical requirements needed to achieve reliable transmission while protecting various primary

services.

As suggested by the term spectrum sensing, with this method, WSD senses the presence of other services using desired channels to determine if there are incumbent user to be protected from interference. Spectrum sensing has the advantage that it does not rely on infrastructure. Spectrum sensing WSD can operate on its own.

There are practical limitations and challenges employing spectrum sensing in WSD. Spectrum sensing would not be able to detect passive services. Also spectrum sensing can be used to maximum effect often only when the characteristics of the primary radio communication is known. A standalone WSD with spectrum sensing technology will need to cover the entire TVWS spectrum with designated detection sensitivity level, which will render the antenna directional instead of omni-directional. The WSD is subject to internal and external noises during sensing. For example, a WSD is likely to be used in an environment with heavy UHF signal such as a living room with TV. In a urban deployment scenario, spectrum sensing also need to address the hidden-node problem. This occurs when WSD's height is lower than protected service devices such as rooftop TV antenna. When blocked by surrounding buildings, WSD might be unable to detect operational incumbent services to be protected due to high attenuation. This will result in WSD transmission causing unwanted interference to protected services. Moreover, separate sensing algorithms are needed for the respective detection of broadcasting and PMSE services.[9]

Based on the findings, ECC report 159 concluded that a WSD using spectrum sensing technology alone is not sufficient to meet the protection requirements.Instead, ECC recommended the use of geolocation database approach. The report also noted that the benefit of applying spectrum sensing to complement geolocation database can be explored further. In line with ECC recommendation, FCC also excluded spectrum sensing from WSD mandatory requirements in 2010.

The geolocation database is a spectrum access management system that assists WSD in selecting their operational frequencies based on geographical frequency availability stored in the database and location information received from WSD.[9]. Geolocation approach requires the exchange of location information as well as protection criteria. WSDs determine their locations and make use of a geolocation database to acquire spectrum for transmission. WSD should communicate its geographical

location, elevation, the accuracy of the location and device information. Based on information given, geolocation database assigns WSD transmitting spectrum and maximum allowed power, as well as the validity period of the grant after which a new request should be placed by WSD.

In beacon approach, local beacon broadcast signals indicating either certain channel is vacant or occupied. ECC and CEPT suggested beacon approach, with its broadcast signals, will introduce interference into TVWS channels as well as consume bandwidth available. Therefore beacon approach has been generally dismissed as viable.[8]

2.4 Spectrum Sharing Approaches in Practice

Introduction of new spectrum sharing scheme, illustrated in Figure 9, will see numerous shared access application occupying designated band, making further refarming or relocation of the spectrum exponentially cumbersome and costly. Therefore any new sharing scheme has to be planned in future-proof manner holistically from several perspectives such as interference, quality of service, fairness of sharing based on tier and sequence of entrance.

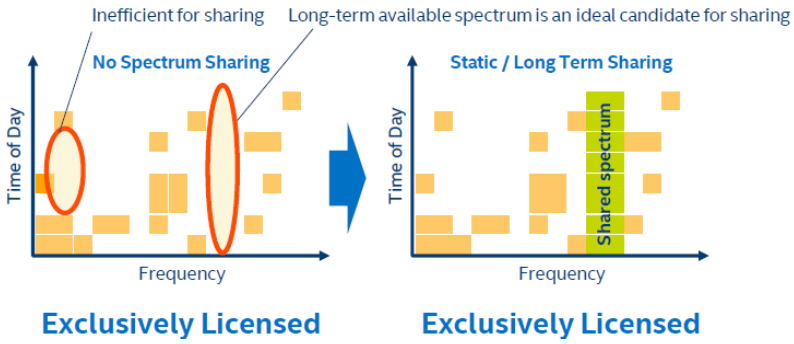

Figure 9: Spectrum Sharing [62]

RSPG suggested[13] that spectrum scarcity happens mostly in urban area where transmitter density is high. Higher frequency bands can achieve frequency reuse in relatively shorter distance while offer greater bandwidth. To mitigate risks of unwanted interference as well as causing issues for future entrants, comprehensive compatibility study should be carried out by the industry and market prior to deploying a new application. Regulators need to balance between market demands for low entry barriers and the needs for spectrum harmonization, long term certainty and

stability of access. Regulation and methodology should be provided in a transparent manner.

2.4.1 TVWS in UK

In 2011, Ofcom issued statement[14] outlines its intention to allow White Space Devices (WSD) to access the TV white space providing that no harmful interference is caused to existing services. Ofcom stated its preference of a harmonized WSD approach across Europe while stating the next steps to establish a geolocation WSD framework within UK. In the same year, a consortium of leading technology companies and organizations conducted a trial of implementing geolocation in TVWS in Cambridge, UK on several use scenarios. The primary purpose of the trial is to assist Ofcom in formulation TVWS regulatory approach and sharing the findings. [15]

The trial demonstrated that it is possible to utilize TV white spaces spectrum in a number of applications, such as rural broadband, urban broadband and machine-to-machine communication. Subject to transmission parameters and regional geometries, the TVWS bandwidth available during the trial was around 120-160 MHz. The trial demonstrated that geolocation database potentially is practical and reliable way of providing frequency sharing control based on location. Geo-location database can provide necessary protection to licensed services and accommodate PMSE usage. In addition, the trial suggests that spectrum monitoring will compliment geolocation database with real time data of frequency usage.[16]

Ofcom led a TVWS pilot running 2013 - 2014 in phases of WSDB contracting and qualification, white space trials, Framework testing and co-existence testing. From Ofcom's perspective, the trial work proved that the proposed framework is feasible in practice. Meanwhile, the trial also showed that WSD technology still requires further development in several areas, such as database qualification and operation, as well as device configurability.

In 2015, Ofcom issued Implementing TV White Space statement, taking geolocation spectrum sharing framework one step further into practical applications. The statement marks Ofcom's decision to go ahead with spectrum sharing access of TVWS 470 to 790 MHz, followed by continuous concept development for several years.[12]

The proposed license-exempt framework with co-existing services is illustrated in Figure 10. In this framework a master WSD will select a geolocation database from one listed by Ofcom. It will communicate with selected database to retrieve relevant transmission geometries and pass it on to slave WSD.

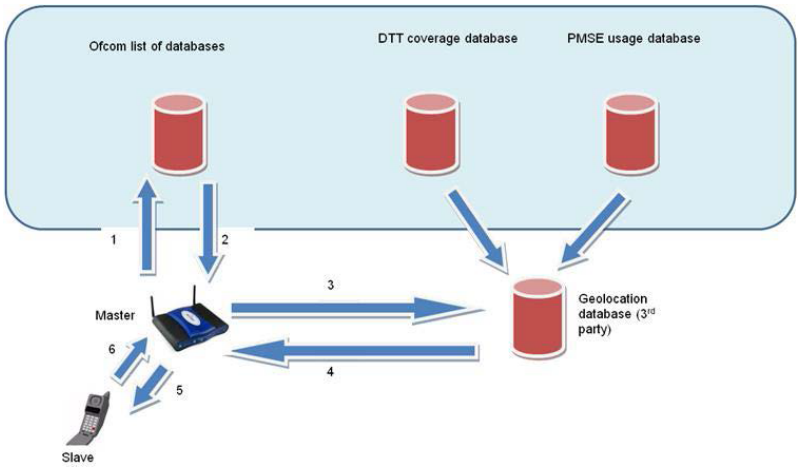

Figure 10: Ofcom TVWS framework[12]

WSDs are tiered as master or slave devices. Master WSD is capable of communicating with geolocation databases. The actual usage of a master WSD may vary depending on the deployment scenarios. Slave WSD operates only based on parameters set by a master device.

The geolocation database will utilize basic operational dataset provided by Ofcom. The dataset includes:

- DTT Coexistence data: maximum allowed WSD transmit power at each 100m x 100m pixel in UK.
- Location Agnostic data: constrains related to PMSE-specific channel 38 usage regardless of WSD location
- PMSE data: location specific licensed PMSE usage
- Unscheduled Adjustments data: allowed transmit power limit update on add-hoc basis

A master WSD will look up a suitable geolocation database(WSDB) to submit its device state to acquire corresponding operational parameters. WSDB will respond to master WSD after calculating based on geographical spectrum availability dataset

and WBD information. Slave WSDs will receive operational parameter broadcast from master device and is allowed to request individual operational parameters as well.

2.4.2 TVWS in US

In 2008, FCC has authorized WSD non-licensed access to TVWS bands in US, provided incumbent services such as television broadcasting and other radio operations are protected from interference via a TVWS geolocation database system. FCC TVWS regulations has since undergone few amendments completing the framework.

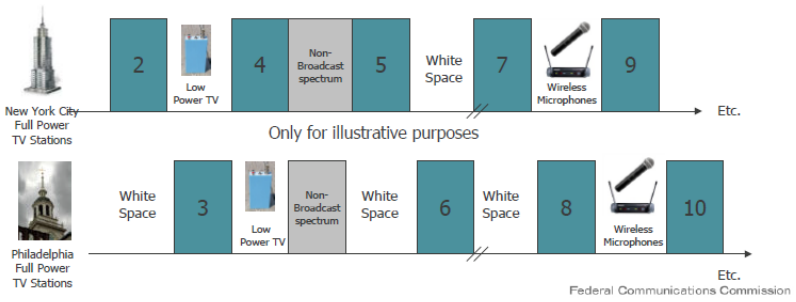

Figure 11: TVWS in US [26]

FCC designated three TVBD(TV band device)/WSD types:

- Fixed: access point or base station type usage, professionally installed and registered in geolocation database.
- Personal/Portable Mode I: end user device type usage, access TVWS via Mode II device
- Personal/Portable Mode II: access point type usage, has GPS capability, provides TVWS access to Mode I device

Spectrum sensing is not mandatory WSD feature by FCC regulation. Instead, FCC reserved two channels exclusively for PMSE usage[22]. It is however mandatory for a WSD to register to geolocation database and complete registration process before transmission. WSD provides its location to the database to retrieve a list of available channels for transmission. Each channel has a bandwidth of 6MHz. WSDs are also required to check in with database periodically to refresh available and granted channel information. WSDs are not allowed to transmit in outlined Protection Zones.

WSD should also be capable of accessing database hosted on the Internet, that identifies incumbent licensed operations entitled to interference protection.[26]

FCC delegates administration of geolocation databases to third party service providers such as Google. By the end of 2013, Google opened up its TVWS spectrum database to the public via the launch of a developer API. This API is also commercially available for WSD equipment manufacturers to register their products to access TVWS.[27] Though there has been commercial application of TVWS technology providing broadband services, overall adaptation has been slow. As spectrum utilization is dense in urban area, availability of TVWS is limited. The main TVWS business deployment scenario in the US is rural broadband.

2.4.3 Licensed Shared Access

In 2011, Radio Spectrum Policy Group (RSPG) report outlined spectrum sharing approach utilizing geolocation database from governance perspective, targeting to facilitate new emerging spectrum sharing regulatory models and providing the European Commission with analysis and further developments concerning Collective Use of Spectrum and other spectrum sharing approaches. The report covers the need of spectrum sharing, the impact and challenges of the sharing approach, the possibility of a regulatory framework and spectrum sharing application on White Space.[13]

RSPG proposed Licensed Shared Access (LSA) as a spectrum sharing regulatory model with a geolocation approach. LSA can be applied to a number of frequency bands on European scale. Under this concept, the initial incumbent licensed user(s) utilising spectrum for a specific application would have to share spectrum with one or several new users for the same, or a different, application in accordance with a set of conditions to be defined through regulation imposed on both the initial user(s) and the new user(s). Static or dynamic licensing conditions will apply on national level.

The report concludes that essential requirement for using white spaces are their sufficient availability and the possibility to identify its users. Geo-location approach can be applied to control the permitted emission levels of equipment as well as to prevent the devices from transmitting at all if necessary, despite the devices are license-exempt.

In 2013, RSPG provided its opinion on spectrum regulatory and economic aspects of LSA[16]. RSPG suggests LSA is about facilitating a more efficient use of spectrum in frequency bands assigned (or expected to be assigned) to one or more incumbent users by introducing additional licensed users: "A regulatory approach aiming to facilitate the introduction of radiocommunications systems operated by a limited number of licensees under an individual licensing regime in a frequency band already assigned or expected to be assigned to one or more incumbent users. Under the Licensed Shared Access (LSA) approach, the additional users are authorized to use the spectrum (or part of the spectrum) in accordance with sharing rules included in their rights of use of spectrum, thereby allowing all the authorized users, including incumbents, to provide a certain Quality of Service (QoS)".

In other words, LSA is a framework under which multiple applications can coexist on same spectrum range based on priority of access and other agreed parameters. The report described the parties involved under a LSA setup.

The administration plays a central role in the introduction of LSA. It should be responsible of LSA framework development, both as regulator as well as facilitator. The administration should define the technical and operational parameters of the shared spectrum access as well as the licensing process, providing certainty to incumbents while offering entry possibilities to additional LSA applications.

The incumbent user refers to current spectrum use right holder, usually government or military organizations. An incumbent is expected to either with own initiative or driven by market condition to set up LSA arrangements governed by the administrations.Naturally incentives should be present for incumbent to adopt to a LSA setup. Those incentives could be lower of total cost of spectrum, more efficient use of spectrum, guaranteed interference protection, as well as financial compensations.

New LSA users are the prospective additional licensees, who will share the spectrum under certain operational conditions. Driven by market, new LSA users are expected to proactively search for sharing possibility and submit proposals. Based on the arrangements, new LSA users will apply for legally-binding licenses from the administration outlining the sharing rules. The availability of spectrum should be provided to new LSA users in transparent and concise fashion, so they can gauge if

there is sufficient commercial incentive to invest and operate under LSA setup.

RSPG is of the opinion that LSA could act as a key enabler to unlock access held by incumbents to spectrum and enable the timely availability of harmonized spectrum for mobile broadband and other applications in a more efficient and actively facilitated approach.[16]

2.4.4 Citizens Broadband Radio Service

In 2015, FCC established a new Citizens Broadband Radio Service for shared wireless broadband use of the 3550-3700 MHz band. This is made possible by accessing 150 MHz spectrum band previously occupied primarily by US military and federal radio services[28]. The introduced CBRS is governed by a three-tier spectrum sharing authorization framework of Incumbent, Priority Access and General Authorized users, accommodating a variety of commercial uses.

Incumbent Access users include authorized federal and grandfathered Fixed Satellite Service users currently operating in the 3.5 GHz Band. These users will be protected from harmful interference from lower tiers. The Priority Access tier consists of Priority Access Licenses (PALs) that, through bidding, will be assigned a 10 megahertz channel within the 3550-3650 MHz portion of the band in a single census tract for three-years. Applicants may acquire up to two-consecutive PAL terms in any given license area during the first auction.[28]

The General Authorized Access tier is unlicensed general access. General Authorized Access users are permitted to use any portion of the 3550-3700 MHz band not assigned to a higher tier user and may also operate opportunistically on unused Priority Access channels. CBRS utilizes dynamic geolocation spectrum access database similar to US TVWS setup. CBRS devices are required to register to a geolocation database access system (SAS) to secure channel grant for transmission.

Based on the operating spectrum, CBRS is more likely for a small cell application comparing with TVWS. At the same time, CBRS and US TVWS system architecture shares similar geolocation mechanism and spectrum sharing philosophy.

2.4.5 Other Spectrum Sharing Approaches

Besides TVWS, LSA and CBRS technologies reviewed above, there are other spectrum sharing approaches which aim at improved efficiency of spectrum utilization and ability to handle increasing data traffic.

LTE Licensed-Assisted Access

Introduced as part of 3GPP R8 standard for 4G, LTE radio devices and network equipments have been on the market since 2008[32]. Supporting both FDD and TDD, LTE is enabled by technologies such as OFDM, SC-FDMA and MIMO. Compared with previous generation of mobile networks, LTE provides significantly higher peak data rate and spectrum efficiency. LTE mobile networks has been widely adopted around the world. As illustrated in Figure 12, LTE technologies evolved over each releases, adding/improving features such as femtocell, carrier aggregation, Self Organized Networks and Heterogeneous Networks. [34]

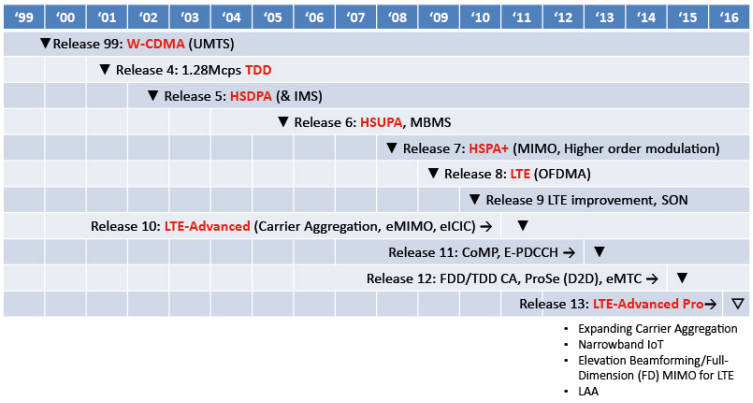

Figure 12: 3GPP Release Roadmap[35]

Delivering service and user experience in licensed spectrum remains 3GPP's top priority. Meanwhile complementary opportunistic use of unlicensed spectrum is regarded of high importance to meet growing demand of mobile data traffic. By 3GPP Release 13, two traffic offload options were introduced, LTE/WiFi aggregation and LTE over unlicensed[30]. While LTE/WiFi interworking framework has been developed since 3GPP Release 8, 3GPP discussion on LTE over unlicensed spectrum was initiated in 2014. The goal was to develop global solution of LTE over unlicensed 5 GHz band, illustrated in Figure 13, capable of coexistence with WiFi and other LTE services by Licensed-Assisted Access operation.

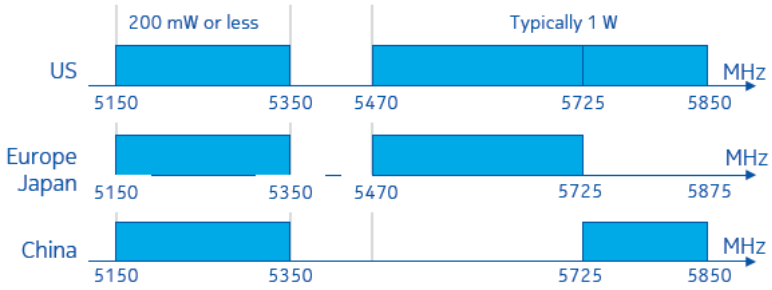

Figure 13: Unlicensed Spectrum Availability Above 5 GHz[29]

There are several aspects where LTE over unlicensed is more advantageous. Being the extension of existing technology, it integrates with existing LTE network infrastructure well. LTE over unlicensed is expected to function as a secondary cell under primary one, providing downlink offloading and boosting data rate. However by nature LTE signals are transmitted without the awareness of surrounding, over the top of other signals sharing the band. Therefore to achieve co-existence and fair sharing, it is important to implement a Listen-Before-Talk mechanism in LTE over unlicensed, much alike WiFi services today. In this way, all neighbouring services can take turn in transmission as well as selecting less-congested bands, to avoid creating a data traffic jam therefore reducing overall transmission capacity.[36]

Operating under Listen-Before-Talk principle, LTE over unlicensed will select most vacant spectrum therefore minimizing interferences in most deployment scenarios[29]. 3GPP also determined that LTE over unlicensed should be able to select carrier and frequency dynamically. On a macro level, both LTE over unlicensed as well as WiFi services would both need to adhere fair sharing mechanism to ensure coexistence. IEEE has been working with 3GPP to shape such sharing mechanism into international standard.[37]

Supplemental Downlink on UHF

As mentioned earlier, mobile network traffic is expected to continue to increase significantly. Users are consuming high-definition video content on the internet via mobile network. This results in an asymmetrical downlink traffic volume over uplink, to a scale of 10:1.[31]

Enabled in the HSPA+ and LTE-Advanced standards, a supplemental downlink(illurstrated in Figure 14) uses unoccupied spectrum to enhance the downlink capability of mobile

broadband networks by aggregating the usual mobile network cell downlink with a supplemental downlink channel(s). Using a wider downlink channel enables faster download speeds for mobile or portable wireless devices and support for a greater number of users.

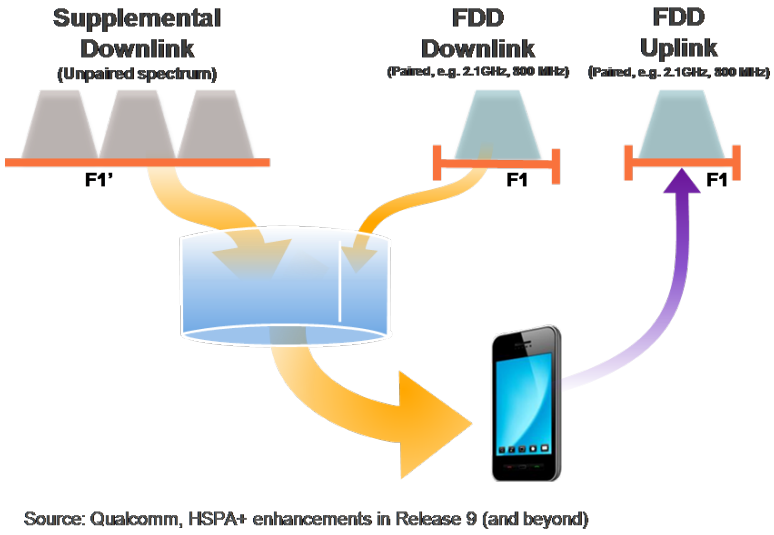

Figure 14: Supplemental Downlink [38]

Currently, there are particular interests in network convergence in the UHF bands. To address the challenge of OTT content providers, Television broadcasters need to be able to offer video on demand service and deliver content also to mobile devices. While traditional DTT lacks video on demand capability and fixed network lacks mobility, LTE broadcast MBMS (Multimedia Broadcast Multicast Service) fulfill both needs.[39]

MBMS provides broadcast multimedia services through the LTE network combining unicast (PDSCH) and multicast (PMCH) services in the same LTE domain. A LTE supplemental downlink at UHF band will benefit both DTT and Mobile broadband service providers. Such downlink can be a part of broadcasting service as well as complementing primary LTE mobile network downlink throughput.

Infrastructure Sharing

As mobile network traffic becomes data-dominant, global telecommunication service providers see a divergence between revenue and cost. As illustrated in Figure 15, the infrastructure cost of faster connectivity and pressure in competitive pricing

are increasing, while revenue generated from mobile data traffic yields diminishing economy of scale.[42]

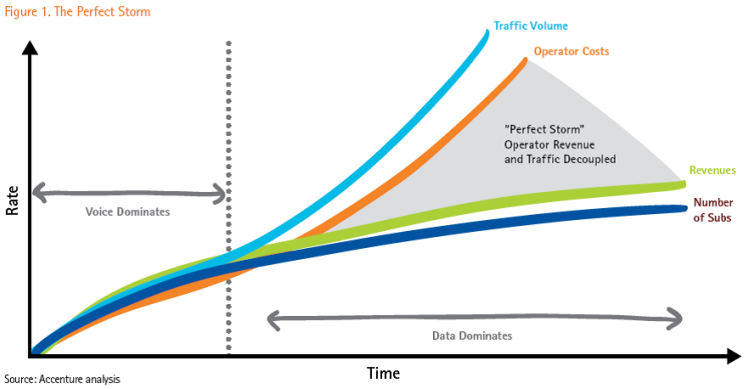

Figure 15: Divergence in Telecommunication [42]

Sharing telecommunication infrastructure is an efficient way for MNOs to reduce upfront investment and improve operational flexibility. Infrastructure sharing is also usually preferred for new MNO entrants, to gain access to licensed spectrum as well as reduce time to market. Technically, network sharing helps MNOs to obtain more efficient spectrum utilization by sharing sites with better coverage and upgraded hardwares[40]. Passive infrastructure sharing shortens network deployment cycles by utilizing existing sites. On the other hand, a Mobile Virtual Network Operator improves efficiency from a business and market perspective, boosting competition and quality of service. A MVNO usually has its own core network and purchases a Mobile Network Operator's data access wholesale for reselling to end customers.[41]

2.5 Summary

In this chapter the very enabling basis of spectrum sharing technology, software defined radio was introduced. The ever-increasing demand of mobile data traffic was examined, along with the notion and possible resolutions of the shortage of spectrum. TV White Space was reviewed as the pioneer of spectrum sharing application. The principles of white space device and TVWS access methods were explained. Starting from TVWS practices in UK and US, this chapter examined spectrum sharing technologies of significance in practice. Licensed Shared Access and Citizen Broadband Radio Service were briefly introduced as they are to be analyzed in more detail in the following chapter. Spectrum sharing approaches evolves on existing mobile network

infrastructure and standards, including LTE over unlicensed, supplemental downlink on UHF and network infrastructure sharing, were also discussed.

3 Architectural Comparison of LSA and CBRS

As one of the first spectrum sharing regulatory framework put into practice, TVWS faces challenges such as conservative approach in protection of DTT and PMSE services, uncertainty in guaranteed QoS as well as the general availability of TVWS spectrum in densely populated areas[44]. Despite the regulatory bodies drove for framework definition and trial runs, the commercial success of TVWS is yet to mature.

With the lessons learnt from the introduction and related operational findings of TVWS, a second generation of spectrum sharing technology is being developed. ETSI and CEPT in Europe have released Licensed Shared Access framework, while FCC in the US introduced Citizens Broadband Radio Service(CBRS). The frameworks currently operate on different bands, but targets to cover all applicable bands as global standards. These systems are expected to act as key components in the next generation mobile network architecture.

Both LSA and CBRS offers more clear business cases comparing with TVWS framework. LSA system makes spectrum available via a two-tier access model, ensuring longer term QoS certainty for potential licensed LSA users (such as MNO) while protect incumbent usage. It extends cellular capacity in Europe by providing 3GPP LTE network operation possibility in 2.3-2.4 GHz band on licensed sharing basis. CBRS offers one additional level of access: Incumbent, licensed/auctioned Priority Access and unlicensed General Authorized Access. CBRS regulations are optimized for small-cell applications, but also accommodate point to point communication especially in rural areas. Small cell and spectrum optimization technologies are expected to advance under CBRS framework.

3.1 Spectrum Sharing Framework

RSPG defines LSA as a regulatory and spectrum management approach which facilitates co-existence of incumbent and limited number of licensees operating in the same band, via an individual licensing regime. This is a complimentary licensing model aiming for mobile network operators to gain shared spectrum access to designed bands, as alternative to traditional spectrum refarming. The shared spectrum access is gained via individual dialogue and licensing, therefore QoS level should be predictable based on the agreement.[50]

The LSA concept supports three sharing methods. LSA licensees and Incumbents will have exclusive access to the spectrum on a given time and location. Static sharing is a time-independent sharing agreement which gives licensee exclusive location-based spectrum access. Semi-static sharing is time-dependent and based on pre-defined parameters. Dynamic sharing assigns bands to licensee based on spectrum availability utilizing geolocation database populated by incumbents.[53]

- Incumbent: current licensed users of the spectrum, can sub-license spectrum to LSA licensee(s)
- LSA Licensee: under sharing agreement operates radiocommunication service within the spectrum, typically a Mobile Network Provider.

CBRS, on the other hand, utilizes a three-tier priority access system for sharing the designated 3550-3700 MHz band with incumbents. This includes a general opportunistic spectrum access layer which is excluded from LSA framework. CBRS' PAL access layer provides the census tract licensing option in which licensee gain location-based access to a 10 MHz channel for three years.[46]

- Incumbent: Current users of the spectrum, mostly military and federal. They are protected from interference from lower tiers.
- Priority Access (PA): This is a licensed mid-tier, which has priority over lower tier but have to concede the spectrum to incumbent users when needed. PA is licensed in 10 MHz unit up to 70 MHz for a period of three years. The licensed spectrum will stay as static as possible while remain dynamically assigned to accommodate narrow-band incumbent scenarios to maximize service continuity.
- General Authorized Access (GAA): This is unlicensed tier which is available for general CBRS accesses and receives no interference protection.

3.2 System Design

The designated LSA operating spectrum is 2.3-2.4 GHz. ETSI developed LSA technical system architecture containing several major components. A LSA operational environment is illustrated below in Figure 16.

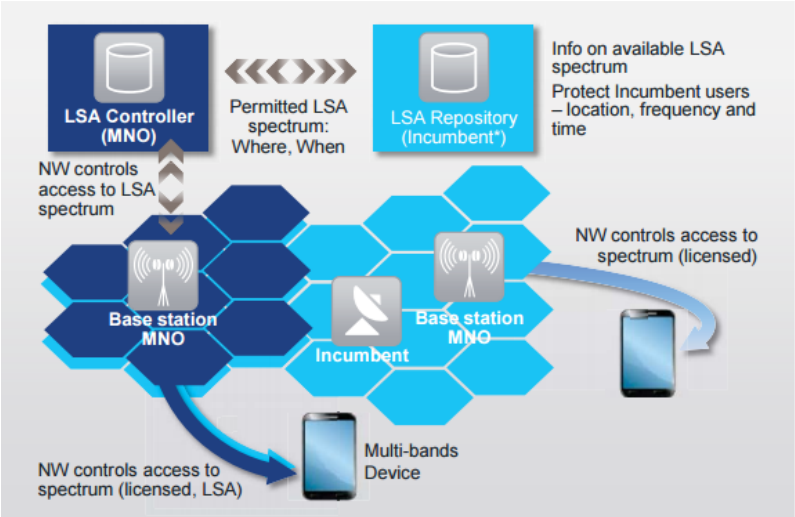

Figure 16: LSA Architecture[54]

LSA Repository (LR) ensures that the operation of LSA system confirms regulation framework and licensing arrangements. It provides registration and information storage of incumbents and relevant protection criteria. LR communicates with authorized LSA Controllers regarding spectrum availability and usage. LR also provide means for National Regulatory Authority (NRA) to monitor LSA system operation.[45]

LSA Controllers (LC) is located within LSA licensee's domain providing spectrum resource availability, allocation request and acknowledgement services. LC communicates with LR as well as LSA licensee's mobile or fixed communication networks. LC therefore provides request processing and decision making capabilities. Together, LSA Repository and Controller provide full functionality of the system, illustrated below in Figure 17.[45]

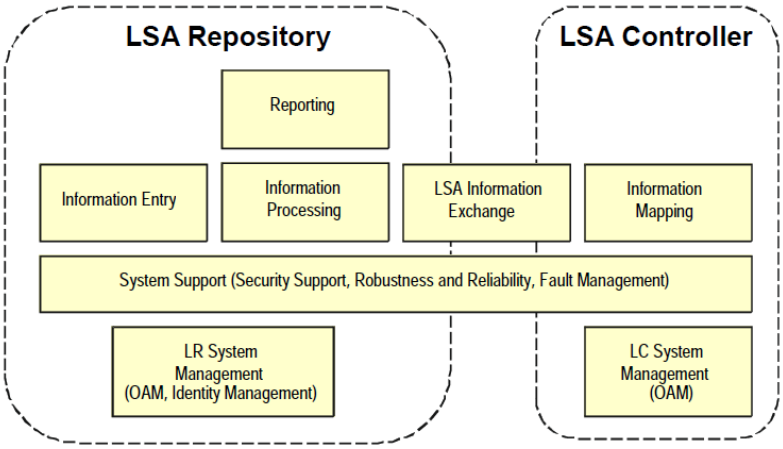

Figure 17: LSA System Functionality[44]

High-level CBRS architecture is illustrated below in Figure 18. Spectrum Access System (SAS) entity is similar to Controller component in LSA, however it is not required to reside within a Mobile Network Operator domain. Based on FCC rules, CBSD are required to register to SAS with their transmission parameters and geometries. Based on the information provided, SAS assign CBSD channels to access and moderate maximum transmission power level, ensure incumbent protection and quality of service. CBSDs are not allowed to transmit before authorized by an SAS.

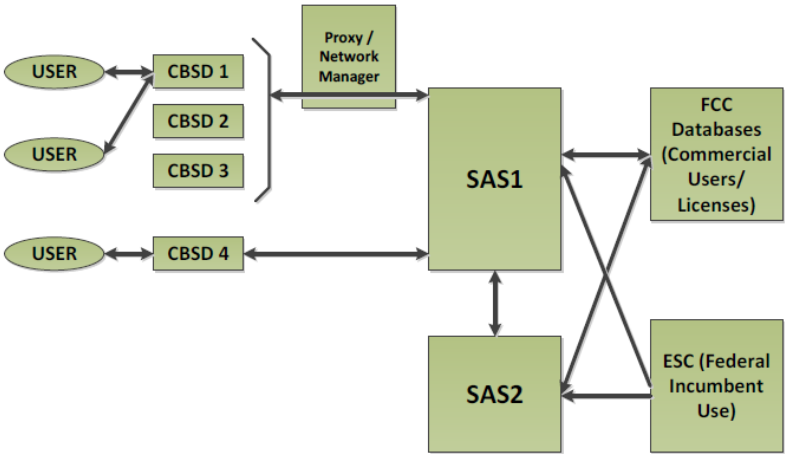

Figure 18: CBRS System[44]

CBRS is different from LSA from several notable technical aspects. In addition to the common Incumbent and licensed tiers, CBRS accommodates an unlicensed GAA tier, which does not exist in LSA. The primary incumbent of CBRS, the Department of Defense, will not disclose spectrum usage of military radar and aviation equipments to geolocation database. Therefore sensing technology, which is complimentary in

LSA, will play a key role in CBRS incumbent protection.[44] CBRS will contain Environmental Sensing Capability components, consisting of networks of sensors, that will detect the presence of incumbent spectrum usage.

3.3 Incumbent Protection

LSA System interfaces between Incumbent and LSA Licensee to enable changes in the LSA spectrum available to be communicated to the Licensee. LSA incumbent protection, illustrated in Figure 19, is primarily achieved by operating based on information available in LSA repository database. LSA database is capable of applying multi-tier spectrum assignments. The LSA database stores protection parameters as well as associated restrictive zone settings listed below.[47]

- Exclusion zone: geographical area within which LSA Licensees are not allowed to have active radio transmitter, normally applicable for a defined frequency range and time period.
- Protection zone: geographical area within which Incumbent receivers will not be subject to harmful interference caused by LSA Licensees' transmissions. Protection parameters are defined using specific measurement quantities and thresholds, applicable for a defined frequency range and time period.
- Restriction zone: geographical area within which LSA Licensees are allowed to operate radio transmitters, under certain restrictive conditions

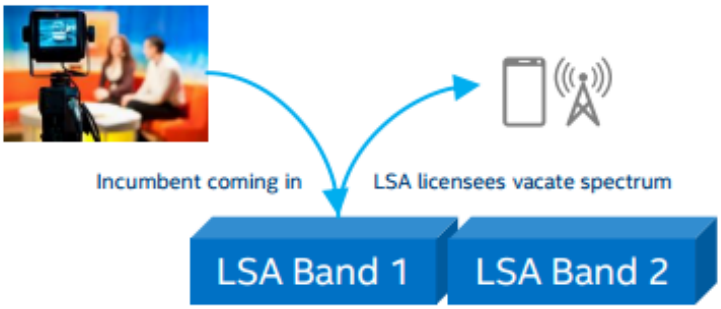

Figure 19: LSA Incumbent Protection [62]

The operational assumption of LSA is that licensees are allocated to large geographical area and most likely function as a MNO. Therefore complex inter-licensee interference protection is not mandatory[44]. On the contrary, neighbouring service

protection is required in CBRS. CBRS PAL are licensed by 10 MHz channels per census tract which is the basic population and geographical unit. Measures to eliminate interference between PALs are needed especially in densely populated urban environments as census tract's size varies.[44]

Incumbent of CBRS band includes military ship-borne radar and naval navigation in coastal areas, military ground radar spread across US in military bases, Fixed satellite service receiving stations and wireless broadband services between 3650 to 3700 MHz. The wireless broadband service incumbents will transit to CBRS PA or GAA layers after five years.[58]

FCC introduces CBRS operation in two phases. Phase 1 will cover inland area of US without ESC capability. Incumbent radar, especially shipborne, will be protected by coastal Exclusion zone prohibiting CBRS operation in 3550-3650 MHz. CBRS operation above 3650 MHz is possible except for fixed CBSDs in vicinity of incumbent radio services. [61] ESC will be implemented as part of CBRS phase 2, covering remainder of US. Exclusion zones in Phase 1 will be converted into Protection zones, where CBSDs operate with grants from SAS connected with ESC.

SAS dynamically assigns and maintains CBRS spectrum use in real time, and there will be no fixed spectral location for PA or GAA allocation. Citizen broadband radio service devices (CBSD) must be able to determine their geographic coordinates every 60 seconds and report any changes in its position to the SAS [55] CBRS monitors incumbent military radar activity via Environmental Sensing Capability. ESC relays detected incumbent activities to SAS which as illustrated in Figure 20 updates CBSDs' transmission parameters to avoid interfering incumbent services.

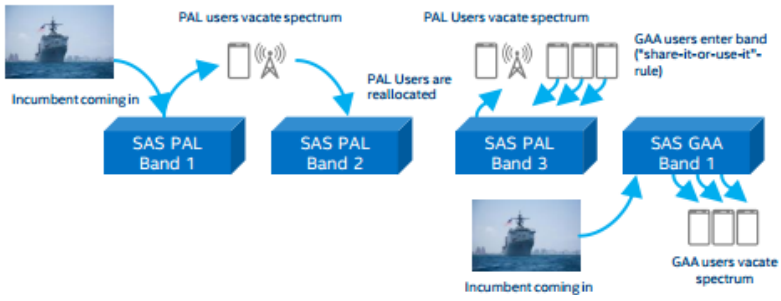

Figure 20: CBRS Incumbent Protection[62]

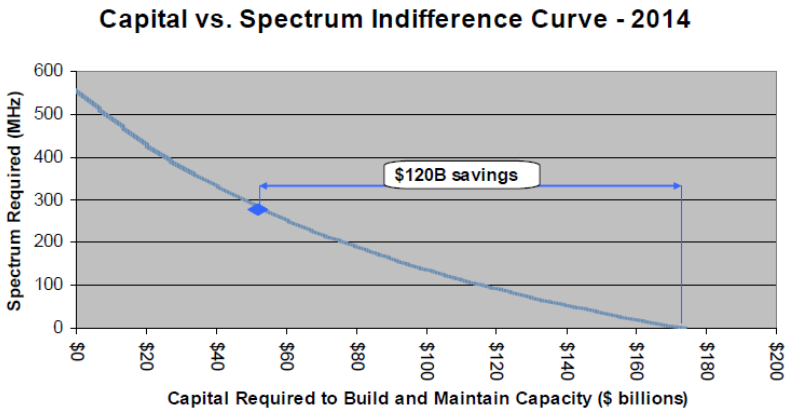

Figure 21: Capital vs Spectrum [19]

3.4 Use Cases

As consumer demand for more mobile data bandwidth is increasing at a rate of close to doubling annually[20], communication service providers need to respond with more capacity. Mobile networks operates with a balance between the amount of infrastructure and spectrum used. Therefore to add network capacity to a given mobile communication technology platform, the service providers and regulators need to choose between increasing network density through the addition of cell-sites or introducing access to new spectrum. Figure21 illustrates that based on FCC estimate, compared to capacity gain achieved by improving the mobile data coverage of densely populated areas with additional in-fill cell stations, introducing additional 270 MHz spectrum for mobile traffic will save approximately 120 billion dollars.[19] Both LSA and CBRS are introduced to extend mobile network capacity on a complementary basis. Functioning as technological and regulatory platforms providing incumbent co-existence mechanisms, they provide investment certainty for relevant business models monetizing shared spectrum access.

The main use case of LSA is the extension of mobile capacity below 6 GHz in Europe. It enables MNOs operating 3GPP LTE network in LTE band 40 between 2.3 and 2.4 GHz based on spectrum sharing with incumbents[44]. A LSA sharing contract can be valid for 10 years or longer. It typically covers a large geographical area on a individual licensee basis. Such setup provides long-term certainty and predictable QoS for mobile networks operators, which are required to justify the investment for large scale network infrastructure. LSA operation can be combined with existing LTE network via carrier aggregation setup. It can also provide downlink offload for primary LTE cells.

Primarily focusing in the US, CBRS shares similar use cases as LSA. One major difference comparing with LSA is the size of the PAL. The licensee is auctioned in 10 MHz lots per census tract and a maximum band of 70 MHz can be allocated to CBRS PA tier users, which are typically MNOs. Remaining spatiotemporal empty spectrum is available to unlicensed users in the GAA tier. It is worth noting that both LSA and CBRS architecturally are spectrum-independent despite of current operational frequency band allocated[44]. For example in the US, CBRS and TVWS geolocation spectrum sharing approaches defined by FCC are rather identical.

LSA and CBRS operations are attractive since they provides possibilities for offloading of the primary mobile network traffic while not interfering with WiFi devices, unlike other similar technologies such as LTE-U. In their designed small/femto cell applications, enterprises can utilize them to build enterprise networks carrying better voice signal and integrates into regular LTE mobile networks. [64] Complementing the mobile network, spectrum sharing is expected to become integral enabler of 5G technology use cases, providing mobile broadband and IoT to consumers as illustrated below in Figure 22.

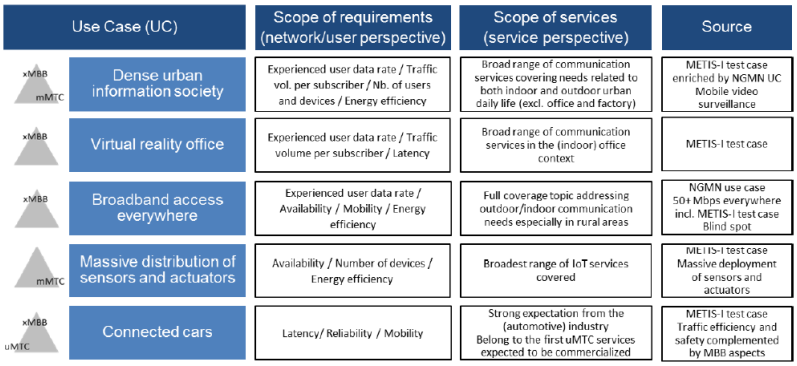

Figure 22: 5G Use Cases[65]

4 SAS-CBSD Protocol

4.1 CBRS Functional Architecture

Spectrum Access System (SAS) authorizes and manages the use of spectrum in Citizens Broadband Radio Service. Citizens Broadband Radio Service device (CBSD) are fixed station or network of stations operates on PA or GAA access tiers in CBRS, providing network accesses. End user devices are not considered as CBSD.

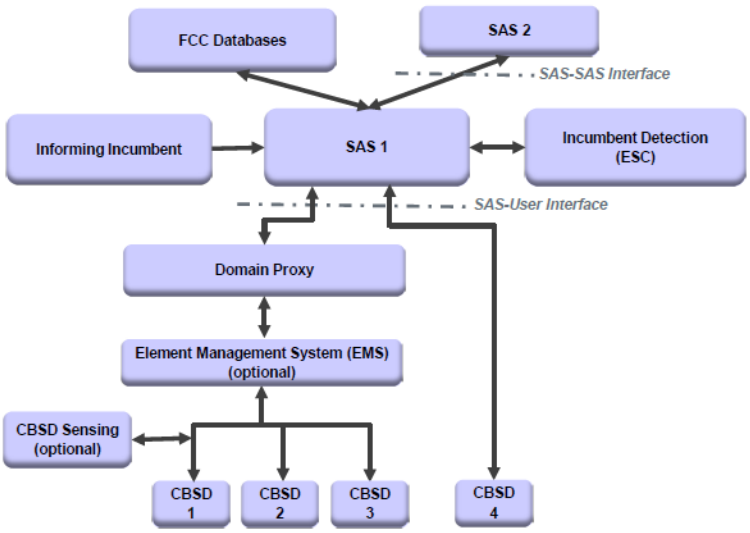

Figure 23: CBRS Functional Architecture[58]

A SAS in the CBRS architecture illustration above enforces and operates CBRS administration policies. It determines the available frequencies at a given geographical location as well as the maximum allowed transmission power level of a CBRS device (CBSD). SAS authenticates CBSDs in its vicinity. It registers and assigns channels to CBSDs. SAS provides incumbent protection based on Exclusion and Protection Zones settings. SAS can also perform incumbent protection dynamically via the assistance of Environmental Sensing Capability (ESC) by sensing the presence of incumbent users.[66]

A Domain Proxy may exist to function as an intermediate network management layer between SAS and a group of CBSDs. It operates CBSD network and manages the network configuration of multiple devices, performing tasks such as bulk CBSD registration and channel assignment.[66]

A CBSD registers itself at SAS with its device and geolocation information. Once

authorized, CBSD transmits within the operation parameters given by SAS and complies with configuration update messages from SAS. CBSD also communicates with SAS reporting channel selected from granted range as well as received signal strength.[66]

As illustrated in Figure 23, CBRS architecture contains SAS-CBSD and SAS-SAS interfaces. SAS-CBSD interface and protocol serves as configuration baseline for compliant user device accesses and operations. SAS-CBSD protocol, which will be reviewed in the next section, accommodates interface operations: SAS discovery, CBSD device registration, CBSD state machine as well as spectrum operation such as request, heartbeat, reassignment, revocation and relinquish. [58] SAS-SAS interface on the other hand enables SAS inter-operability. The interface provides standard protocols for coordination between SASs to facilitate non-interfering CBSD operation administrated by adjacent SAS.[75]

4.2 SAS-CBSD Interface Procedures

This section describes the procedures used in SAS-CBSD communications. SAS-CBSD interface resembles a typical server-client setup. CBSD is required to perform a set of pre-requisite procedures before spectrum request can be granted and actual transmission can commence. CBSD is required to discover SAS occupancy at a given location and register itself. Once registered, CBSD can communicate with SAS to request for spectrum assignment by providing a set of requested parameters and own geometries. SAS evaluates the request based on situational awareness of incumbent presence as well as exclusion zoning. SAS replies CBSD with either granted or rejected request response. If spectrum is allocated, CBSD is required to check in with SAS on periodic basis to refresh/revalidate the spectrum grant. SAS will adjust or revoke the granted spectrum based on updated incumbent presence information. CBSD can also vacate granted spectrum and inform SAS.

4.2.1 Pre-requisite Procedures

Before actual SAS-CBSD communication, prerequisite procedures have to be performed.

- CBRS user registration: As the legal entity which owns or be responsible for CBSDs, CBRS user must register with FCC to create a user id for SAS. Similar as in other service registration processes, CBRS user will need to provide basic

information such as address and contact, as well as acknowledge the terms and conditions of CBRS. Upon completion of user registration, an unique user id is generated to be used in SAS-CBSD communications.

- Priority Access License (PAL) right management and id registration: Each PA licensee should be given unique credential to authenticate at SAS. Once authenticated, SAS will allocate reserved PAL channels based on FCC database search result based on the licensee's credentials.

- Device Type parameters: device parameters based on the manufacturer should be provided by the vendor and entered into CBSD databases.

- Installation parameters: CBSD geometries based on physical installation is not pre-known. Therefore such parameters will need to be logged into SAS database by installation professionals manually.

- Security: Communication security framework is needed so that SAS, CBSD and Domain Proxys can authenticate counterparts during communication setup and procedures.

4.2.2 SAS Discovery and CBSD Registration

CBSD or a Domain Proxy on behalf of an CBSD, discovers SAS during this procedure. CBSD locates a SAS via static and dynamic methods to establish connection for registration. CBSD should then register to SAS with its device information as well as communication geometries in order to request for spectrum allocation.

4.2.3 CBSD Spectrum Request and Relinquish

Based on current proposal, to start the procedure CBSD will initiate a spectrum inquiry request to SAS, providing PAL ID if applicable. SAS will provide available channel information to CBSD in spectrum inquiry response. CBSD then sends a grand request with the channel selected to SAS.

SAS performed detailed interference assessment on the channel selected. If CBSD's channel selection can be accepted, SAS sends response containing granted frequency, bandwidth and valid duration. CBSD starts transmission with the perimeters given and inquires SAS periodically to validate the grant.

If the connection to SAS is lost, CBSD will assume that existing grant is invalidated and a new spectrum request is needed for channel assignment. If the assigned access is no longer needed, CBSD issues Relinquish request to release the channel to SAS. If the channel grant duration expired, CBSD sends Relinquish request to SAS as well. CBSD can sends another spectrum request to SAS if a channel is still needed.

4.2.4 SAS Spectrum Reassignment/Revocation

Once channel is assigned to CBSD, SAS has the ability to revoke or reassign spectrum to CBSD responding to higher priority access requests. When a higher priority user requires access to the spectrum previous assigned to CBSD, SAS will notify that CBSD should release the channel. An alternative channel may be given, if available, as part of the notification.

4.3 CBSD State and State Transition

There are multiple operation states of a CBSD, such as Registered, Granted and Transmission.

- "Registered" state indicates that a CBSD has gone through registration process and provided its device information at SAS. Once CBSD is in Registered state, it is ready to submit grant request to SAS with desired operating parameters.
- "Granted" state indicates that CBSD has been granted transmission right by SAS with operating parameters. CBSD or the domain proxy responsible of it will start sending heartbeat requests to SAS maintaining communication. SAS will provide operational instruction update via its heartbeat responses.
- "Transmission" state indicates that SAS confirmed transmission parameters in its heartbeat response to CBSD. CBSD is allowed to transmit. If SAS suspends/revokes granted operating parameters due to detection of incumbent, CBSD will move back to Granted state.

CBSD operational state diagram is as follows.

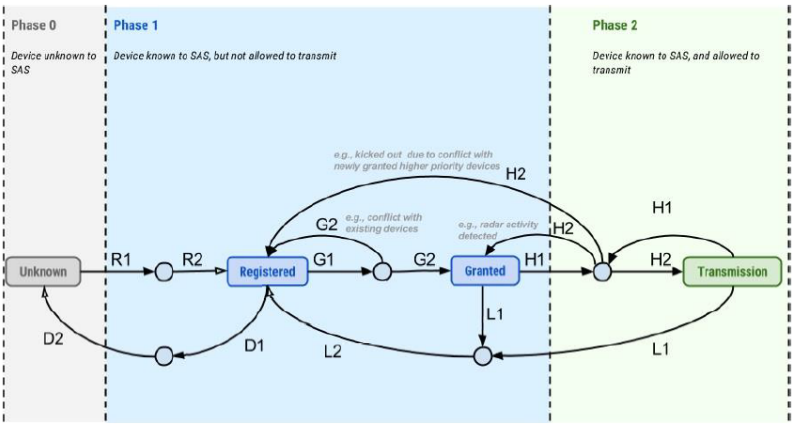

Figure 24: CBSD Operational State[68]

As shown above, a CBSD transits between states via interface messaging with SAS. Simulation of SAS-CBSD interface is covered in the next chapter.

- R1: Registration Request. Sent as initial registration request when a CBSD is in Unknown state and wants to transmit in SAS controlled spectrum. The request will timeout if response is not received within expected time.
- R2: Registration Response. Contains SAS's response to R1 request. SAS will either approve or reject the request. If rejected, error message will be included. If approved, CBSD will be in Registered state.
- G1: CBSD Grant Request. Contains CBSD's intended operational parameters for SAS's approval.
- G2: SAS Grant Response. SAS's response of CBSD's grant request. If approved, CBSD transits in to Granted state with the duration of the grant as well as the heartbeat interval parameters set in G2 message.
- H1: Heartbeat Request. Periodically CBSD checks in with SAS via heartbeat message to ensure SAS is aware that CBSD is active based on the grant given.
- H2: Heartbeat Response. Allows SAS to periodically confirm, suspend or deny a grant. In this way SAS can adjust spectrum occupancy when a higher-tier device such as incumbent moves into its proximity. As long as the grant is confirmed, CBSD is allowed into Transmission state
- L1: Relinquish Request. Sent by CBSD when the grant given is not used anymore. It contains CBSD ID and operation state.

- L2: Relinquish Response. Grant termination confirmation from SAS. Once terminated, the CBSD moves back to Registered state.
- D1: Deregister Request. Sent by CBSD to de-list itself from SAS when it moves out of SAS' jurisdiction zone, determined by maximum allowed distance to SAS. This request is also sent when a CBSD is decomissioned.
- D2: Deregister Response. SAS' confirmation of CBSD's deregister request.

5 SAS-CBSD Interface Simulation

A simulation environment is built to validate SAS-CBSD interface and message design outlined in SAS to CBSD Protocol Technical Report[68]. In the simulation, SAS and CBSD instances go through communication processes and transit between stages. The goals of the simulation are:

- implement end-to-end SAS-CBSD communication flow, validate SAS-CBSD interface message structure and info model(e.g. CBSD data structure) during CBSD state transitions
- demonstrate basic geolocation functionalities
- implement basic incumbent protection determination mechanisms.

5.1 Simulation Setup

The simulation execution is powered by Robot Framework. Robot Framework is an open source test automation framework under Apache license. It is Python-based and utilizes Keyword Driven testing approach, suitable for automating processes and constructing reusable modules.[76] There are a wide variety of Python function modules (test libraries) available which can be plugged into Robot Framework for added capabilities. The simulation environment used is a Windows machine.

5.1.1 Robot Framework Installation

Robot framework can be run as a standalone JAR distribution. However the most organic way to utilize Robot Framework is by installing it on top of Python. Prerequisite of Robot Framework installation is Python 2.7 runtime environment. The installation of the Framework and its libraries can be done by using python installation manager(pip).

Robot Framework provides extensive functionalities via packaged libraries. Remote library is needed for the simulation. This library is provided as a part of the standard installation. It will be used for the running of python servers. Without a client - server setup we will only be able to run synchronous single thread execution without the ability to e.g. maintain parallel communication sessions in asynchronous mode.

5.1.2 Keyword-Driven Automation

The central concept of automation functionality construction in Robot Framework is Keyword[77]. Each keyword en-capsules reusable functionality of performing either a singular or combined action, such as "Send" or "Receive and Reply". An automation scenario can be built from keywords logically by combining and reusing the functionalities, in the same way how object oriented classes and objects can be used.

Keywords for a certain common topic are bundled into Test Libraries. A Test Library can be referred or inherited and customized in the same way as a Class in Object Oriented programming. Customized keywords can be stored in Resource file. Keywords are executed with needed parameters and initialization in a Test Script file. A test script captures one or multiple automation scenarios. Such scenarios can be stored in Test Case file, in the same way Keywords are stored in Test Library. Test Case files can be referred by Test Script, when execution of the stored scenario is needed. Figure 25 shows keyword driven automation script used in the simulation.

```
*** Settings ***
Library           OperatingSystem
Library           Remote    http://${ADDRESS}:${PORT}    WITH NAME    SASServer
Library           Remote    http://${ADDRESS}:${PORT2}    WITH NAME    CBSD1
Variables         Variables/variables.py

*** Variables ***
${ADDRESS}        127.0.0.1
${PORT}           8270
${PORT2}          8272

*** Test Cases ***
0.CBSD1 Initialization
    CBSD1.Initialize CBSD    CBSD_a1

1.CBSD1 Unknown->R1->R2->Registered
    CBSD1.Send R1 Request
    SASServer.Listen And Respond R1
    CBSD1.Receive R2 Response

2.CBSD1 Registered->G1->G2->Granted
    CBSD1.Send G1 Request    CBSD_OpParam1
    SASServer.Listen And Respond G1
    CBSD1.Receive G2 Response

3.CBSD1 Granted->H1->H2->Transmission
    CBSD1.Send H1 Request
    SASServer.Listen And Respond H1
    CBSD1.Receive H2 Response

4.CBSD1 Transmission->L1->L2->Registered
    CBSD1.Send L1 Request
    SASServer.Listen And Respond L1
    CBSD1.Receive L2 Response

5.CBSD1 Registered->D1->D2->Unknown
    CBSD1.Send D1 Request
    SASServer.Listen And Respond D1
    CBSD1.Receive D2 Response
```

Figure 25: Keyword-Driven Simulation

5.1.3 Automation Script Layout

The simulation script as well as resource files can be viewed and edited from Robot Framework's IDE RIDE. Once RIDE is installed and Python environment variable were set, it can be launched by typing "ride" in command prompt.

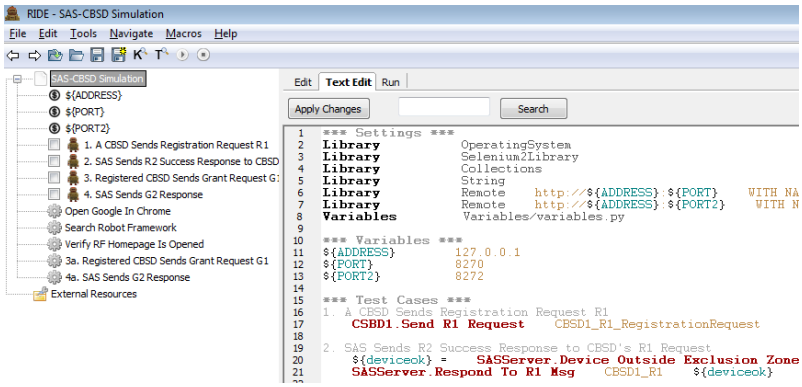

Figure 26: RIDE User Interface

RIDE's user interface is shown in above Figure 26. The left pane contains navigation point inside the script. The text editor is on the right. Besides Text Edit, Edit and Run tabs host variable configuration and script execution controls.

Robot Framework automation scripts, like the one displayed in the screenshot, contain several sections.

The first section is **Settings**, where Libraries, Variable files and Resource files are declared. Robot Framework installable contains pre-packaged libraries. External libraries should be installed via own installable or via Python Package Manager pip. The script will refer to customized variables and functions defined specifically for the simulation where are stored in separate files. These file names and their relative path to the automation script are included in Settings section.

The second section in the script is **Variables**, where in-script variables are defined. The difference between this section and the Variable files in the previous section is that the variables here are defined for this script only and can only be used by this script. This allows easy configuration updates to simulation parameters such as URLs and IP port numbers. If the variables are to be reused or referred elsewhere from another simulation script, it is always possible to move the variable to its own file and add the file in the Settings section.

The third section is **Test Cases**. This section contains the simulation scenarios and steps to be executed in succession. The simulation steps are always constructed from Keywords stored either in a Library, in a resource file, or in the next section of the script. It is always good practice to encapsulate functionalities in Keywords to ensure

re-usability and reduce script update effort required. However it is also possible to define the step from the most basic libraries if required. The simulation steps can be configured via parameter settings, subject to the construction of the keyword used.

The fourth section is **Keywords**. Script-specific keywords are defined here for reference in the Test Case steps. Similar to variables, keywords defined in this section cannot be used elsewhere. Keywords from Libraries and Resource files included in Settings can be used. This section should contain only script-specific customizations or grouping of keywords.

5.2 Simulation Core Modules

5.2.1 SpectrumAccessSystem

SpectrumAccessSystem modele contains SAS-CBSD communication functionalities. Each SAS or CBSD device is an instance of SpectrumAccessSystem. This module contains functionalities such as interface message construction and interpretation and geolocation determination. Key functions are listed below.

- Device Outside Exclusion Zone: return True if device is outside given exclusion zone
- Device Distance To Incumbent: calculate CBSD distance to incumbent
- Device Pathloss To Incumbent Sufficient: determines if device pathloss is suffcient
- Send SAS Message: sends interface message
- Load SAS Message Template: load interface message template
- Listen And Respond G1 Esc: SAS functionality, listen and respond G1 request, perform geolocation determination based on incumbent location
- Listen And Respond G1 Excl Zone: SAS functionality, listen and respond G1 request, perform geolocation determination based on Exclusion zone coordinates
- Initialize CBSD: CBSD functionality, intialize CBSD based on data file provided
- Send R1 Request: CBSD functionality, sends R1 request.

5.2.2 SASLocationService

SASLocationService module contains geolocation and incumbent protection utility functions. The functions defined in SASLocationService module are accessed by SpectrumAccessSystem module, providing overall geolocation determinations.

- Device Distance To Incumbent Km: returns distance between CBSD and incumbent given in km.
- Device Distance To Zone Km: returns distance between CBSD and Exclusion zone given in km.
- Device In Zone: determine based on coordinates given if CBSD is within given zone.
- Get Distance Km: utility function, gets distance between two points in km by applying haversine formula.
- Pathloss Km Mhz: utility function, calculates flat fading pathloss based on given frequency in MHz and distance in km.

5.2.3 SASDatabase

In order to keep track of spectrum usage by CBSD and incumbent as well as the grant status, SAS will employ database. SAS database is simulated by reading and writing JSON files generated and stored on a predefined location. Database function utilizes meta class files to generate data objects for CBSD, grants and spectrum usage. In the simulations, SAS and CBSD instances share the same database by accessing JSON files stored on defined locations.

- Meta: stores meta data objects
- Init: stores initialization data objects
- Operation: stores data objects used during system operations

5.2.4 CBSD Meta Data

CBSD Meta data serves as template to construct CBSD data object dynamically to store and track CBSD status as well as grant provided. This meta structure is not defined in SAS-CBSD interface specification. It is constructed as part of the implementation of the simulation, based on SAS-CBSD interface messages relevant for CBSD status as listed below.

- fccId: defined in registrationRequest message
- cbsdCategory: defined in registrationRequest message
- callSign: defined in registrationRequest message
- userId: defined in registrationRequest message
- airInterface: defined in registrationRequest message
- cbsdManufacturer: defined in registrationRequest message
- cbsdSerialNumber: defined in registrationRequest message
- maxNumberOfGrant: defined in registrationRequest message
- sensingCapability: defined in registrationRequest message
- installationParam: defined in registrationRequest message
- error: defined in registrationResponse message
- grant: defined in grantResponse message
- operationState: defined in heartbeatRequest message
- operationStatusReq: defined in heartbeatResponse message

5.3 Simulation Implementation

CBSD transits between states following SAS-CBSD communication protocol. Different data contents are transmitted as SAS-CBSD interface messages between each state. To simulate the interface communication, an automated environment is built in which virtual SAS and CBSDs are communicating to each other based on the interface specification. Major SAS functionalities are simulated as well, such as incumbent protection and geolocationing.

The simulation repository consists of:

- Functional Modules: contains SAS and CBSD functionalities, such as communication via interface messages and geolocation determination. They are used as building blocks for SAS and CBSD instances. SAS server and CBSD simulators are launched as Remote Servers instances. In this way, multiple devices can be operational at any given time.

- Message Templates: contains empty interface messages used as template during communication simulation. Data format is defined in communication protocol documentation.

- Message Storage: SAS-CBSD communicated messages are stored temporarily. This simulates both actual system storage as well as over the air communication before it reaches the destination. In simulation, predefined data format/templates are stored as JSON files. They are stored as single layer or multi-layer objects. SAS and CBSD instances will load the data templates via standard JSON interface when needed from predefined location. Tx instances simulate outgoing communication by packaging the data content and template and saving the data object as a JSON file in predefined location. Rx instances simulate receiving communication by fetching messages are a given stage from predefined location.

- Scenario Script: An automation script contains detailed communication process steps. It directs the initialization of virtual devices as well as the communication process to be executed.

- Result Illustration: To illustrate simulation result, once geolocation determination is completed the result is saved as graphic file into predefined location for reference.

5.3.1 CBSD State Transition

SAS server and CBSD device instances are launched as Remote Server from command prompts. The instances are running in parallel based on configuration stored in respective python files. SAS-CBSD interface messages are implemented in JSON as specified[68] for the simulation.

Initialization

To begin simulation, python remote servers should be designated as SAS server or CBSD device. The instances can be launched from command prompt manually. It is also possible to create a batch file to initiate multiple instances at the same time. Each remote server instances is defined in a python file, where the instance name, IP address and port are defined. In this simulation, each SAS or CBSD instance is derived from module SpectrumAccessSystem.py, containing system operation functions. In Robot Framework automation script, each Remote Server instance is referenced in Library section. In below Figure 27, A remote server instance is derived from SpectrumAccessSystem module, with name CBSD1, operating on IP 127.0.0.1 and port 8272. The instance was launched from command prompt.

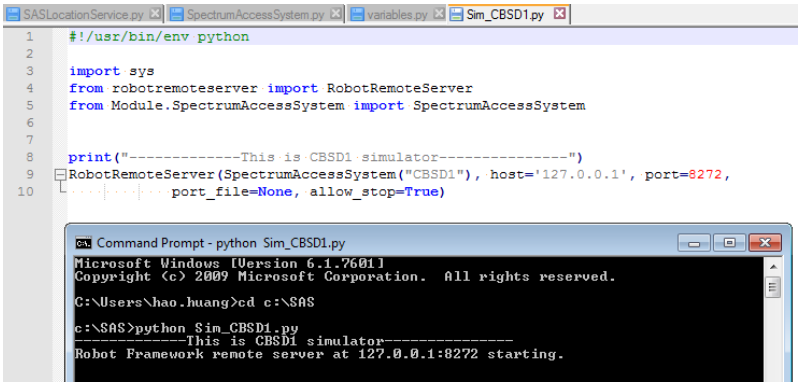

Figure 27: Remote server configuration and command prompt launch

As the first step of the simulation, initialization parameters are loaded into CBSD instance from data file. This is achieved by calling keyword Initialize CBSD defined in SpectrumAccessSystem, illustrated in Figure 25.

```
0.CBSD1 Initialization
    CBSD1.Initialize CBSD    CBSD_a1
```

A CBSD id is given to the instance. The instance stores the id as its FCC Id and retrieves initialization file based on the id. Below Figure 28 shows Python implementation of CBSD initialization function.

```
def initialize_CBSD(self, cbsdId):
    """initialize CBSD paramters by reading from a file"""
    cbsd_obj = SASDatabase().initialize_CBSD_object(cbsdId)
    # save cbsd id to self
    self.cbsdId = cbsdId
    # save CBSD dynamic obj to file
    cbsd_obj["fccId"] = cbsdId
    SASDatabase().save_CBSD(cbsd_obj, cbsdId)
```

Figure 28: CBSD initialization

CBSD Data Model

The initialization file is a derivation of CBSD data model. The data model is not defined in SAS-CBSD interface specification. The model is created for simulation purposes accommodating all relevant CBSD data to be stored. It is an assortment of parameters specified in SAS-CBSD interface protocol[68].

- fccId: The FCC certification identifier of the CBSD.
- cbsdCategory: Device Category of the CBSD.
- callSign: A device identifier provided by FCC.
- userId: The identifier of a CBSD user.,
- airInterface: A data object that includes information on the air interface technology of the CBSD.
- cbsdManufacturer: A unique name for the CBSD manufacturer.
- cbsdSerialNumber: A serial number assigned to CBSD by the CBSD device manufacturer.
- maxNumberOfGrant: The maximum number of grants that the CBSD can use simultaneously.
- sensingCapability: The array describes all sensing capabilities of CBSD relevant to SAS operation.
- installationParam: A data object that includes information on CBSD installation.
- error: This parameter includes information on whether the corresponding CBSD request is approved or disapproved for a reason.

- grant: customized field, contains combined data structure of grant request and response.
- operationState: contains CBSD's operation state
- operationStatusReq: If True, CBSD should include operation parameters and operation state in the next heartbeat request.

Unknown - Registered

By default, CBSD is initialized in state Unknown. The simulation assumes CBSD has discovered SAS. To register to SAS, CBSD will send Registration request (R1). Once request is received by SAS, SAS will review request and accept or reject the request in Registration response (R2).[68]

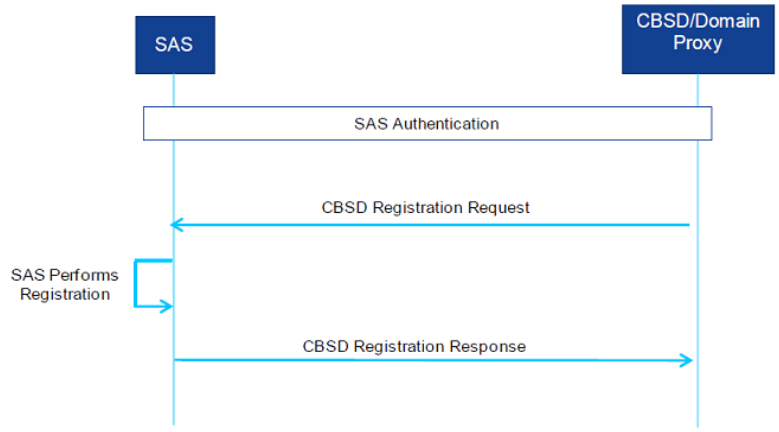

Figure 29: CBSD Registration Process[68]

The simulation of CBSD registration process consists of three steps: CBSD sends R1 request, SAS listens for incoming request and respond, CBSD listens for incoming response and interpret it. Illustrated in Figure 25, these steps are grouped under automation step:

```
1.CBSD1 Unknown->R1->R2->Registered
    CBSD1.Send R1 Request
    SASServer.Listen And Respond R1
    CBSD1.Receive R2 Response
```

The function sending R1 request will first load CBSD data model by id. Then the function loads R1 message template, fills it with CBSD data and sends out the

message by storing the message in Transmission folder. The Transmission folder simulates communication traffic and carries "in the air" interface messages.

The automation script calls listen and respond R1 function at SAS side. The function search for the first available R1 request message parse it. Once the response is determined, the function loads R2 message and populate it based on parsed R1 message and registration determination. It sends R2 message forward by storing in Transmission folder. In the simulation SAS and CBSD shares the same database storing operational data. As specified by FCC[66], CBSD should be able to self-determine and keep track of its own state. Therefore the majority of the state tracking operations are handled on the CBSD side.

CBSD listens and parses R2 response in similar manner. CBSD checks for respond content and possible errors to determine the next step. If the respond does not contain any error, CBSD changes its state to Registered and stores this information into database.

Registered - Granted

Once CBSD is in Registered state, it is entitled to initiate Grant process by sending G1 request. CBSD is able to inquire spectrum availability from SAS via Spectrum Inquiry process. For simplicity, simulation does not cover Spectrum Inquiry process. Alternatively, spectrum availability is preconfigured. It is worth noting that the Spectrum Inquiry process can be simulated in the same way as Registration or Grant processes. The geolocation and incumbent protection aspects of Spectrum Inquiry process are covered in the Grant and Transmission process simulations.

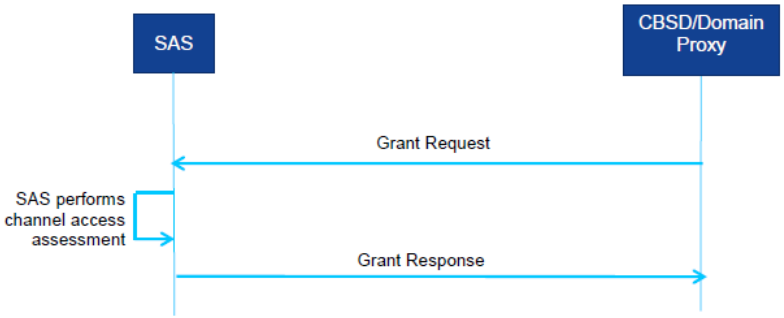

Figure 30: CBSD Grant Process[68]

Grant process simulation is also a three-step approach: CBSD sends G1 request, SAS listens and reply with G2 response and CBSD listens and interpret G2 response. G1 request contains CBSD's operational parameter object. Illustrated in Figure 25, these steps are grouped under automation step:

```
2.CBSD1 Registered->G1->G2->Granted
    CBSD1.Send G1 Request    CBSD_OpParam1
    SASServer.Listen And Respond G1
    CBSD1.Receive G2 Response
```

The operational parameter contains frequency range and peak power requested by CBSD. CBSD is also able to specify if it is requesting GAA or PAL access by providing PAL credentials[68]. In G2 response, SAS will specify heartbeat interval in case the request is granted. Once in Granted state, CBSD is not yet allowed to transmit without completing the Heartbeat procedure.

Granted - Transmission

To complete Heartbeat procedure, CBSD sends heartbeat H1 request referring to the grant id. SAS evaluates CBSD operation parameter in the referred grant against spectrum usage. In response, SAS specifies transmit expire time, heartbeat duration as well as updated operation parameter if needed. Illustrated in Figure 25, these steps are grouped under automation step:

```
3.CBSD1 Granted->H1->H2->Transmission
    CBSD1.Send H1 Request
    SASServer.Listen And Respond H1
    CBSD1.Receive H2 Response
```

Once the heartbeat procedure is completed, CBSD logs itself into Transmission state.[68]

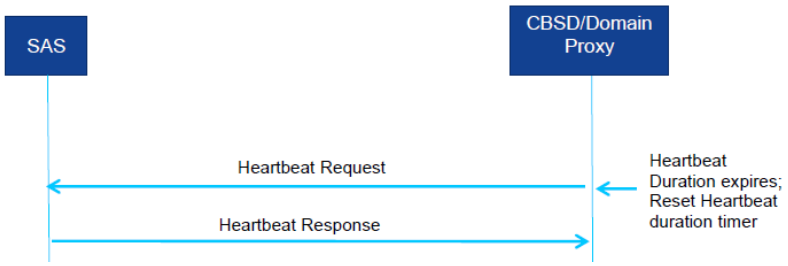

Figure 31: Heartbeat Process [68]

Transmission - Registered - Unknown

When CBSD voluntarily pauses or ends transmission, it can inform SAS accordingly via relinquish request. In simulation, CBSD will populate a L1 request template with its FCC Id and the active grant Id, and then sends the message to SAS for processing.

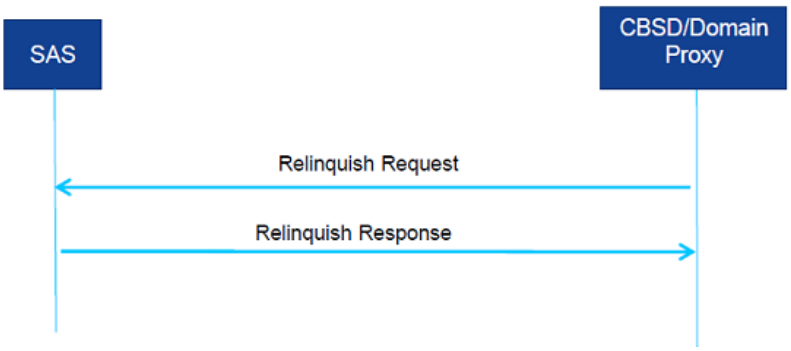

Figure 32: Relinquish Process [68]

When being decommissioned or moved out of SAS' area of jurisdiction, CBSD should initiates a Deregistration D1 request to de-list itself from SAS database. The D1 request contains CBSD's id. The relevant steps illustrated in Figure 25 are:

```
4 CBSD1 Transmission->L1->L2->Registered
    CBSD1.Send L1 Request
    SASServer.Listen And Respond L1
    CBSD1.Receive L2 Response

5.CBSD1 Registered->D1->D2->Unknown
    CBSD1.Send D1 Request
    SASServer.Listen And Respond D1
    CBSD1.Receive D2 Response
```

5.3.2 Geolocation

Spatial Analysis Modules

To simulate SAS geolocationing, Geographical Information System (GIS) functionalities are implemented in SASLocationService module. Two widely-used Python GIS packages, Shapely and Fiona, are imported into the module. Shapely provides

geometric calculation capabilities, such as constructing geometric objects and performing calculations. Fiona on the other hand functions as interface to GIS files, reading from and writing into commonly used GIS data formats.

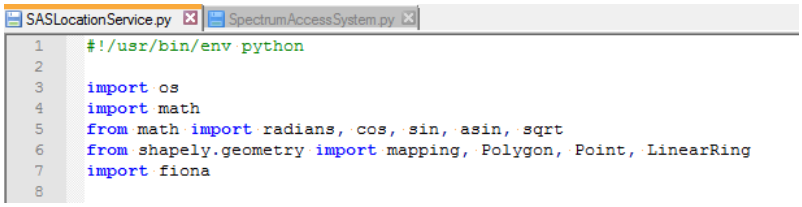

Figure 33: Importing Shapely and Fiona

Shapely is based on Geometry Engine Open Source (GEOS), which provides below core capabilities[70]:

- Geometries: Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon, GeometryCollection
- Predicates: Intersects, Touches, Disjoint, Crosses, Within, Contains, Overlaps, Equals, Covers
- Operations: Union, Distance, Intersection, Symmetric Difference, Convex Hull, Envelope, Buffer, Simplify, Polygon Assembly, Valid, Area, Length

From spatial analysis perspective, CBSDs and Incumbent users can be defined as Points, while exclusion/protection zone can be defined as Pologons. Shapely provides all basic spatial operations needed. For example, to check if a CBSD is within an exclusion zone, the *within* function of Point class can be used. The Point class also provides *distance* function which returns the closest distance between two spatial objects.

```
def device_in_zone(self, devicexy, zone):
    """
        return true if given point is within given polygon. Inputs are
        devicexy: coordinates of device
        zone: coordinates of the zone
    """
    poly = Polygon(zone)
    point = Point(devicexy)
    if point.within(poly):
        return True
    else:
        return False
```

Figure 34: device_in_zone function

Fiona is a Python wrapper for vector data access functions from the OGR library (Geospatial Data Abstraction Library). It focuses on vector representation of discrete

entities in GIS repository.[71] It is not able to process raster data which represents continous fields. Fiona is primarily used in writing GIS output into graphics.

```
def save_point_to_shapefile(self, xy, name, file):
    """
        Save given point coordinates to file. Inputs are
        xy: coordinates of device
        name: description of device
        file: location of shp file
    """
    schema = { 'geometry': 'Point', 'properties': { 'name': 'str' } }
    with fiona.open(file, "w", "ESRI Shapefile", schema) as output:
        point = Point(xy)
        output.write({
            'properties': {'name': name},
            'geometry': mapping(point)
        })
```

Figure 35: Saving point to Shapefile

In above code, Fiona saves the schema given into a ESRI shapefile spatial data format. Shapefile format is recognized by most GIS softwares and is used to store vector data. The simulation provides spatial data output in Shapefile formats so that the coordinates can be presented graphically in GIS software's user interface.

Plotting

Geolocation data is processed in simulation and saved into data files. Plotting functionalities are implemented to better demonstrate the simulation outcome. Simulation plotting functions utilizes Python MatPlotLib module. The module provide Matlab-like 2-D graphic output capabilities. The output can be saved as image files. MatPlotLib is primarily used to plot points, lines, polygons and other shapes processed during the simulation. The simulation also utilizes Basemap MatPlotLib toolkits. Basemap does not directly creat plots. It facilitates the creation of map projections based on the coordinate input given.

Seperate installations of MatPlotLib and Basemap are required. After installation, the modules are included in python modules created for simulation.

```
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.basemap import Basemap
```

Distance Calculation

As the geolocation capability provided by Shapely is by latitude and longitude,

it is necessary to convert a distance by coordinates into kilometers to enable propagation calculation. The conversion from coordinates to kilometers achieved by the implementation of haversine formula. The haversine formula calculates the greater circle distance between two coordinates, as illustrated below. The formula gives rather accurate output even for shorter distances. It is best to provide float as input to the formula for better precision.

$$\begin{aligned} a = &\quad \sin^2(\Delta\varphi/2) + \cos\varphi_1 \times \cos\varphi_2 \times \sin^2(\Delta\lambda/2) \\ c = &\quad 2 \cdot \arctan 2(\sqrt{a}, \sqrt{1-a}) \\ d = &\quad R \times c \end{aligned} \tag{1}$$

where φ and λ are latitude and longitude in radian, and R is Earth's radius with a mean of 6371km.

The correctness of distance calculated can be validated by comparing simulation calculated distance of well-known destinations with the distance calculator and data available on the Internet. The distance between New York and Atlantic City are calculated based on their coordinates:

```
Coordinates:
    AtlanticCity = (-74.4229, 39.3643)
    NewYork = (-74.00594, 40.71278)
Calculation:
    Distance in KM between two coordinates
        ${distance} =    SASServer.Device Distance To Incumbent
            ${NewYork}    ${AtlanticCity}
```

The function gave an output of 154km, which is in line with the publicly available distance between the two cities:

```
INFO : ${distance} = 154.087883792
```

The plotted result is illustrated below in Figure 36.

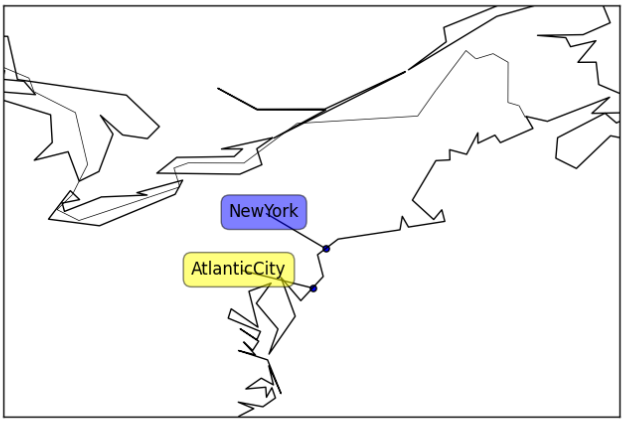

Figure 36: New York to Atlantic City

5.3.3 Incumbent Protection

Basic incumbent protection mechanisms are implemented in the simulation. SASLocationService.py provides CBSD interference calculation by applying free space path loss model, simulating a non-urban communication environment. The function takes operating frequency of CBSD in MHz and distance between CBSD and incumbent user in km. The distance can be computed based on above-mentioned geolocation function based on CBSD and incumbent coordinates. It is possible to take into account of antenna gains at transmitter and receiver.

$$FSPL(dB) = \quad 20log_{10}(f) + 20log_{10}(d) + 32.44 \tag{2}$$

where f is frequency in MHz and d is distance in km, with corresponding constant 32.44.

Based on the calculated power level, SAS can determine if CBSD can be granted to transmit at requested parameters. Figure 37 below illustrates CBSD allowed transmission power can be determined based on pathloss model applied and relative distance to known incumbent. Aegis in the figure represents a seaborne radar operating on a US missile cruiser off the coast.

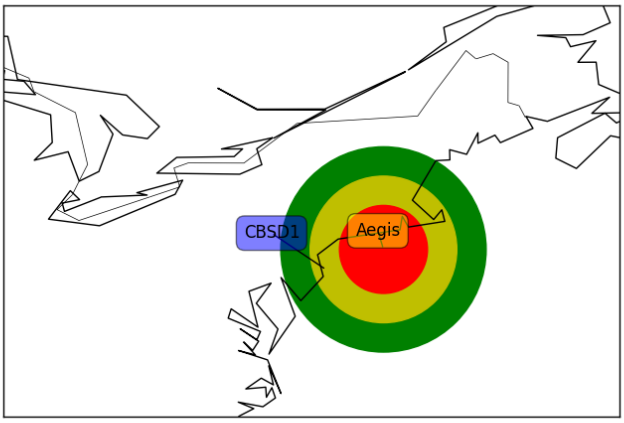

Figure 37: CBSD allowed power in relation to incumbent location

5.4 Simulated Scenario and Results

Outlined in FCC report 15 47A1[66], CBRS incumbent protection will be implemented in phased approach. As recommended by NTIA, the first phase implements static geographic exclusion zones along the coastline and around fixed inland radar sites. CBSD with EIRP of maximum 30dBm in a 10 MHz band is allowed to transmit when outside of the exclusion zones. In the second phase, approved ESC should operate together with SAS, providing an engineering-based dynamic incumbent protection assisted by sensing network. The simulation aims to reflect this phased approach. Each phase is covered in one simulation scenario.

5.4.1 CBRS First Phase Simulation

The first phase simulation covers a scenario where one CBSD registers to SAS and requests for transmission. SAS determines if CBSD's request can be granted based on the location of CBSD in relation to a known exclusion zone.

The Exclusion zone boundary is defined by a series of points in longitude and latitude pairs. SASLocationService functional module will construct the exclusion zone as a polygon by connecting the points. The coordinates are given as fixed parameters.

```
EXCLUSIONZONE = [(-88.63, 40.88), (-88.63, 42.88), (-86.63, 42.88),
```

```
(-86.63, 40.88), (-88.63, 40.88)]
```

CBSD's peak power and operation frequency range are defined in CBSD initialization file. The data model used is derived from CBSD grant request message structure. Based on SAS-CBSD interface specification[68], peak power parameter should be defined in dBm/MHz while operation frequency range is defined as FrequencyRange data object, containing numerical values of low frequency and high frequency in Hz. The parameters given to CBSD are 3 dBm/MHz peak power and 10 MHz band, matching FCC specification.

```
"peakPower": "3",
"operationFrequencyRange":
    {
    "lowFrequency": "3650000000",
    "highFrequency": "3660000000"
    }
```

CBSD coordinates are given in initialization file as well, in installationParam data object.

```
    "installationParam":
        {
        "latitude": "45.88",
        "longitude": "-87.63"
        }
```

CBSD coordinates are provided to SAS in R1 Registration request message. The operation parameters including peak power and frequency band are provided to SAS in G1 Grant request and refreshed per request in H1 Heartbeat request messages.[68] As illustrated in below Figure 38, in this particular scenario CBSD location given is outside of the defined exclusion zone.

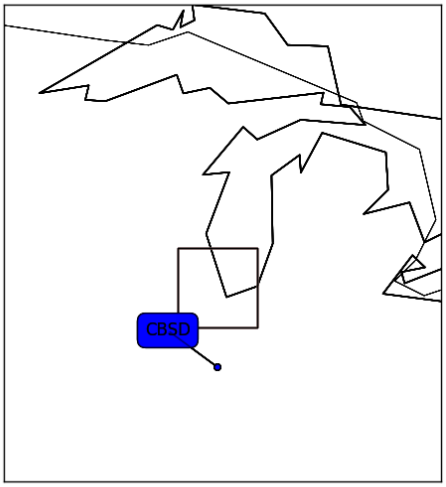

Figure 38: Scenario 1a: CBSD outside Exclusion Zone

Based on location analysis, SAS grants CBSD rights to transmit. CBSD transits from Unknown state to Registered then to Granted. The transitions and related parameter updates are kept in CBSD data object stored in database. THis is illustrated below in Figure 39. CBSD installation parameters and coordinates are populated accordingly. It is noted during the simulation that SAS should have access to CBSD data in order to perform geolocation determination. The G1 grant request message structure specified does not contain installation parameters. Therefore SAS will have to inquire CBSD data based on given CBSD id from SAS database.

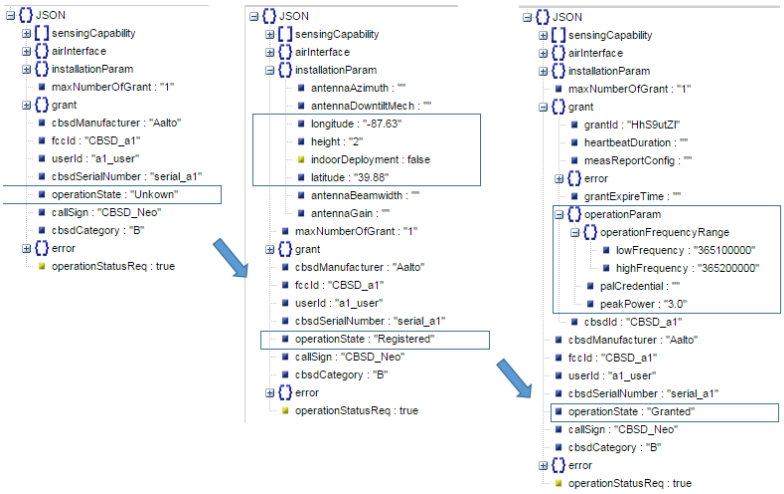

Figure 39: Scenario 1a: Unknown to Granted

The simulation then completes the full state transition cycle until the CBSD has been fully deregistered. The CBSD data objects are illustrated below.

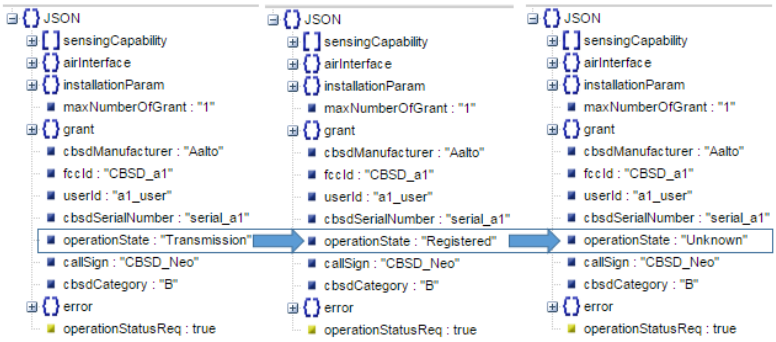

Figure 40: Scenario 1a: Granted to Unknown

When CBSD is located within the Exclusion zone, SAS should not grant CBSD rights to transmit. This is simulated by adjusting CBSD's coordinates:

```
"installationParam":
    {
     "latitude": "41.88",
     "longitude": "-87.63"
    }
```

The relative location of CBSD and defined Exclusion zone is illustrated below.

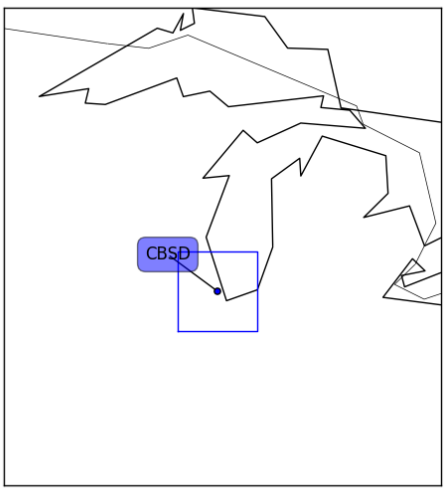

Figure 41: Scenario 1b: CBSD within Exclusion Zone

SAS determines that CBSD is within Exclusion zone during Grant request review. SAS response with error code 400 Interference[68]. CBSD interprets the Grant response and stays in Registered state. The error code is stored in CBSD data object accordingly. This is illustrated below in Figure 42.

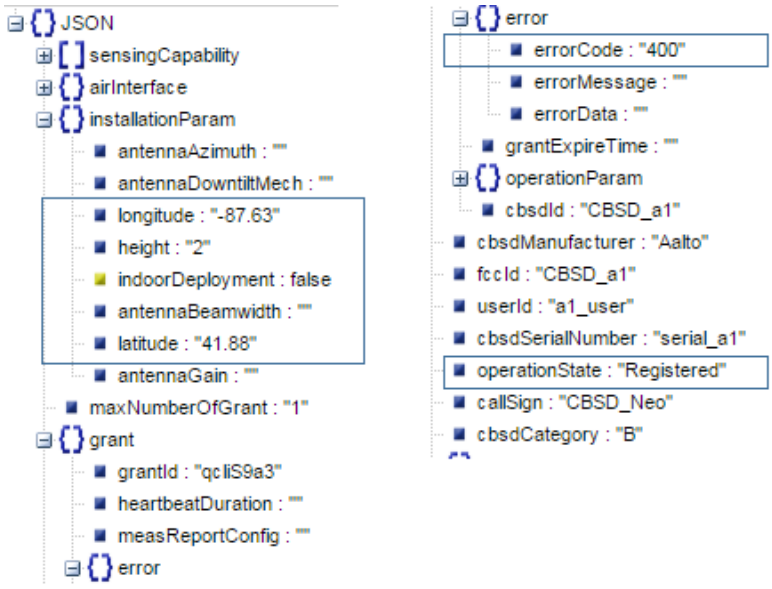

Figure 42: Scenario 1b: CBSD not granted to transmit

5.4.2 CBRS Second Phase Simulation

In the second phase of the CBRS deployment, ESC sensing mechanism will provide an engineering based dynamic access approach for SAS. This means instead of a fixed Exclusion zone, SAS will utilizes ESC sensed incumbent user existence within its vicinity to determine if certain CBSD is allowed to transmit with requested operation parameters.

To simulate the dynamic access aspect of the CBRS deployment, one Incumbent user is defined instead of an exclusion zone. In the simulation, SAS will calculate if the distance between CBSD and detected Incumbent user is further than pre-defined separation distance. CBSD location selected for the simulation is on the coastline while incumbent location is at sea, simulating an incumbent seaborne naval radar system. Free space pathloss model is used along with an arbitary minimium pathloss of 140dB to determine if the separation between CBSD and Incumbent user is sufficient to allow transmission. The pathloss parameter is approximated based on data available on NTIA FastTrack report[21] and SAS-CBSD specification, considering possible frenquency offset and antenna gain.

The CBSD and Incumbent locations are defined as follows:

```
CBSD1 = (-74.0444, 40.6892)
Aegis = (-72.0444, 40.6892)
```

This gives an separation distance of 168 km, illustrated in Figure 43 below.

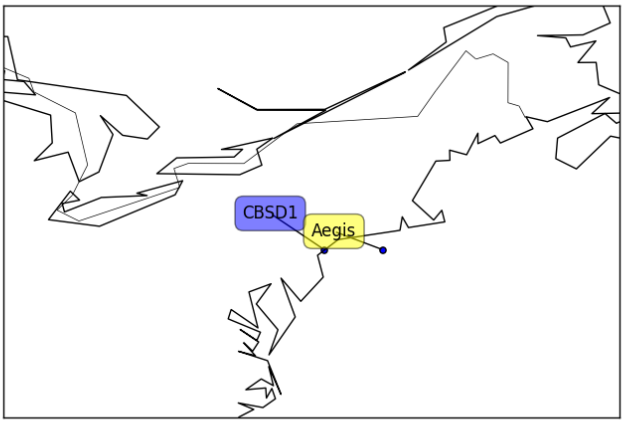

Figure 43: Scenario 2a: CBSD and Incumbent Separation Sufficient

The path loss calculation result is 143.77dB, which is larger than minimal required 140dB. Therefore CBSD request is granted, as illustrated in CBSD data object below.

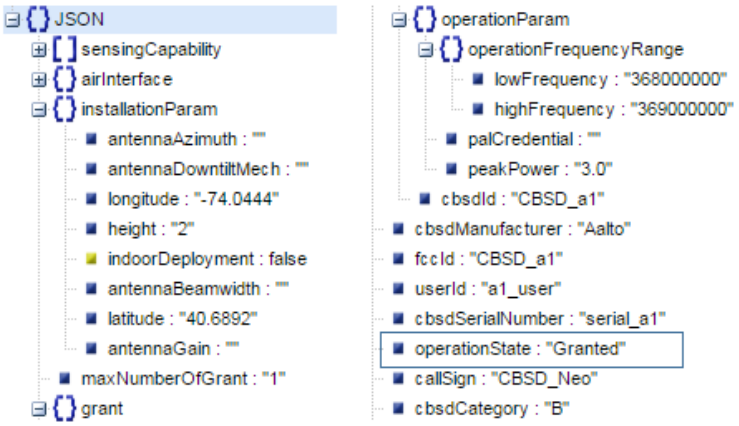

Figure 44: Scenario 2a: CBSD granted to transmit

As the naval radar platform cruises at sea, its distance in relation to CBSD will change. This is simulated with below coordinates.

```
CBSD1 = (-74.0444, 40.6892)
Aegis = (-74.0000, 40.6892)
```

CBSD and Incumbent coordinates are illustrated in below Figure 45.

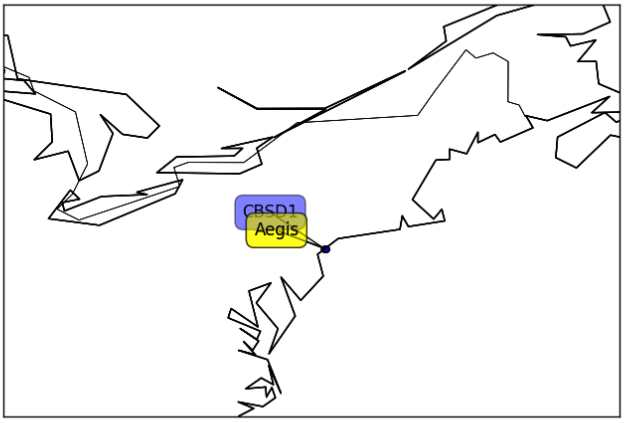

Figure 45: Scenario 2b: CBSD and Incumbent Separation Insufficient

The path loss calculated is smaller than 140dB, therefore CBSD grant request is rejected by SAS with error code 400. This is illustrated in the figure below.

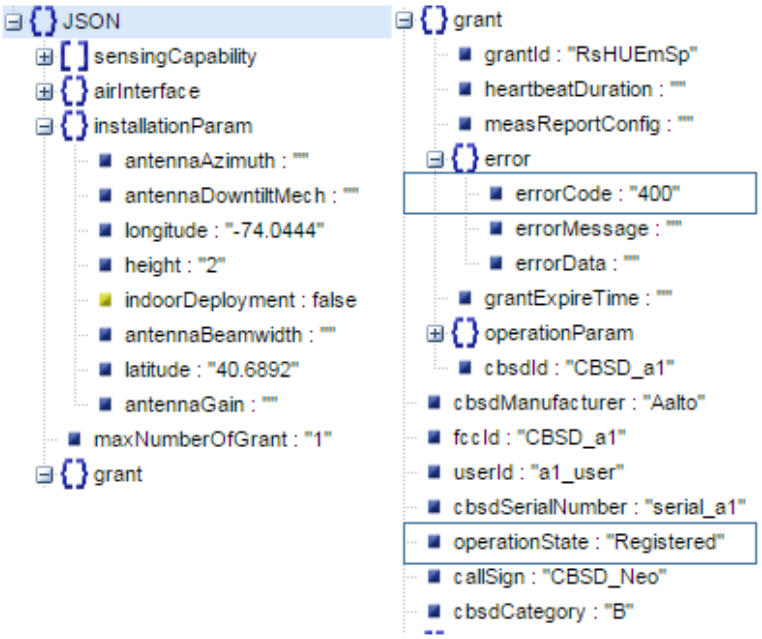

Figure 46: Scenario 2a: CBSD not granted to transmit

5.4.3 Conclusion

The simulation covered both CBRS deployment scenario with fixed Exclusion zone and deployment scenario with dynamic access enabled by incumbent sensing on a network level. The simulation implemented key SAS system components and functionalities, providing modules designated for SAS determination, geolocation and data processing. The exchange of information between SAS and CBSD was implemented. End to end CBSD state transitions were simulated.

Based on simulation outcome and observation, conclusion can be made that SAS-CBSD interface and message structure are working as intended. The message structure supports the exchange of information between SAS and CBSD. However it is worth noting that in order to keep track of CBSD state transition as well as to reference CBSD attributes, a CBSD data model should be defined for SAS database. In addition to CBSD self-determination characteristics, SAS will still have to store information regarding every CBSD in its jurisdiction. A common CBSD data model will help standardizing 3rd-party SAS database APIs.

The simulation assumed straightforward single Exclusion Zone/Incumbent and single CBSD scenarios, with free space pathloss model. Incumbent protection determination

will be more complex in a multi-incumbent and/or urban environment. ESC will provide SAS the needed elasticity in spectrum assignments as demonstrated in the simulation, as CBSD transmission can be granted dynamically based on its relative distance to detected mobile incumbents.

6 Summary

Driven by technological trends and further adaptation of mobile broadband-based communication and data access, the demand for more network capacity is ever increasing. However the projected spectrum crunch is largely caused by current spectrum assignment framework lagging mechanisms to use the spectrum more efficiently. Regulatory bodies such as FCC and Ofcom have reacted by introducing dynamic spectrum sharing access frameworks.

Allocated on UHF bands with prime propagation characteristics, TVWS spectrum are made available for shared access as part of the digital dividend when TV broadcasting switches from analog to digital. Implementing spectrum sharing while protecting incumbent services on TVWS has been thoroughly studied. The studies concluded that geolocation should be implemented as the primary shared access mechanism.

By nature, any spectrum sharing framework will introduce a number of occupants to shared spectrum, making any future reallocation prohibitively costly. The practical implementation of the framework requires future-proof design and extensive field testing. This is evident in TVWS spectrum sharing framework development and trial iterations in the UK and the US.

Based on the lessons learnt from TVWS, the regulatory bodies introduced the next iteration of spectrum sharing frameworks, namely LSA and CBRS. They share several architectural similarities: primarily geolocation based, operates on higher frequency suitable for small-cell implementation and utilizes tiered access approach providing more clear use cases for business adaptation, especially by MNOs. LSA with its two-tier framework is primarily reserved for MNO and can be considered a more dynamic version of traditional spectrum licensing scheme. CBRS with a three-tier access framework, enables additional use cases in its GAA tier.

The primary network control component of CBRS is SAS, providing access control for CBRS user devices, referred as CBSD. A 3rd-party geolocation database provider community was established for TVWS in US. FCC relies on the same community to host geolocation database for CBRS. Geolocation-based spectrum access messages are relayed via SAS-CBSD interface. The interface specification follows typical web interface structure and is frequency-independent. This means the interface specification is compatible for future adaptations of shared spectrum beyond the

band currently made available to CBRS operations.

Similarly to other spectrum sharing frameworks, one of the main focuses of CBRS is incumbent protection. Incumbent protection presents unique challenges for CBRS, as the primary incumbents are US military ground-based and sea-borne radar systems. Such military operation cannot be pre-announced and registered in public geolocation databases.

To address the challenge, FCC adopted a two-phased CBRS deployment approach. Firstly, CBRS is deployed inland to provide maximum coverage with conservative Exclusion zone defined along the US coastline, in which CBSD are prohibited to transmit. In the next phase, Environmental Sensing Capability employing network of sensors listening for unannounced incumbent presence is deployed in CBRS, assisting SAS in spectrum access decision making. As shown in the simulation, ESC provides CBRS engineering-based dynamic incumbent protection capability, extending its business case by making CBRS deployment in most-populated coastal area in the US viable.

Addressing the need to allocate and utilize spectrum more efficiently, spectrum sharing technologies are becoming more industrialized and mature since the introduction. In foreseeable future, spectrum sharing will become an integral part of wide-band mobile network platform, both as technique and as supplementary networks.

References

[1] 4G Americas, "Spectrum Sharing", White Paper, 2014.

[2] Coherent, "Report on enhanced LSA, intra-operator spectrum-sharing and micro-area spectrum sharing", Deliverable D4.1, Tech. Rep., 2016.

[3] Ericsson, "5G Radio Access - Capabilities and Technologies", White Paper, 2016.

[4] ITU-R, "Definitions of Software Defined Radio (SDR) and Cognitive Radio System (CRS)", ITU-R SM.2152, Tech. Rep., 2009.

[5] Jones, E, "Software Defined Radios, Cognitive Radio And The Software Communications Architecture (SCA) In Relation To Comms, Radar and ESM", *ISBN 978-0-86341-939-3*, 2008.

[6] Mitola, J, "Software radios-survey, critical evaluation and future directions", *IEEE Aerospace and Electronic Systems Magazine (Volume: 8, Issue: 4, April 1993)*.

[7] http://mil-embedded.com/articles/evolving-technology-sdr-cognitive-radio/

[8] CEPT, "A preliminary assessment of the feasibility of fitting new/future applications/services into non-harmonised spectrum of the digital dividend", CEPT Report 24, Tech. Rep., 2008.

[9] ECC, "Technical And Operational Requirements For The Possible Operation Of Cognitive Radio Systems In The 'white Spaces' Of The Frequency Band 470-790 MHz", ECC Report 159, Tech. Rep., 2011.

[10] Bazelon, McHenry, "Substantial Licensed Spectrum Deficit(2015-2019): Updating the FCC's Mobile Data Demand Projections", White Paper, 2015.

[11] http://money.cnn.com/2012/02/21/technology/spectrum_crunch/

[12] Ofcom, "Implementing TV White Spaces", Statement, 2015.

[13] RSPG, "Report on Collective Use of Spectrum (CUS) and other spectrum sharing approaches", RSPG11-392, Tech. Rep., 2011.

[14] Ofcom, "Implementing Geolocation - Summary of consultation responses and next steps", Statement, 2011.

[15] Cambridge White Spaces Consortium, "Recommendations for Implementing the Use of White Spaces: Conclusions from the Cambridge TV White Spaces Trial", White Paper, 2011.

[16] RSPG, "RSPG Opinion on Licensed Shared Access", RSPG13-538, Tech. Rep., 2013.

[17] ITU-R, "IMT Traffic Estimates For The Years 2020 To 2030", ITU-R M.2370-0, Tech. Rep., 2015.

[18] PCAST, "Realizing The Full Potential Of Government-held Spectrum To Spur Economic Growth", Tech. Rep., 2012.

[19] FCC, "Mobile Broadband: The Benefits Of Additional Spectrum", Tech. Paper, 2010.

[20] CISCO, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2015–2020", White Paper, 2016.

[21] NTIA, "An Assessment of the Near-Term Viability of Accommodating Wireless Broadband Systems in the 1675-1710 MHz, 1755-1780 MHz, 3500-3650 MHz, and 4200-4220 MHz, 4380-4400 MHz Bands", Tech. Rep., 2010.

[22] http://reboot.fcc.gov/spectrumdashboard/resultSpectrumBands.seam? conversationId=31953

[23] The Broadband Commission, "The State of Broadband 2014: Broadband For All", White Paper, 2014

[24] Ericsson, "Ericsson Mobility Report", White Paper, 2016.

[25] https://en.wikipedia.org/wiki/Last_mile

[26] https://www.ietf.org/proceedings/81/slides/paws-1.pdf

[27] http://www.theverge.com/2013/11/15/5106218/google-database-api-brings-white-space-broadband-closer

[28] https://www.fcc.gov/rulemaking/12-354

[29] Nokia, "LTE for unlicensed spectrum", White Paper, 2014.

[30] Flore, D, "3GPP & unlicensed spectrum", IEEE 802 Interim Session, 2015.

[31] Digital Europe, "White Paper On Supplemental Downlink In The UHF Band", White Paper, 2014.

[32] http://www.3gpp.org/technologies/keywords-acronyms/98-lte

[33] https://www.iith.ac.in/ tbr/teaching/docs/LTE-Tutorial.pdf

[34] http://www.unwiredinsight.com/2012/3gpp-lte-releases

[35] http://www.3gpp.org/about-3gpp/about-3gpp

[36] http://www.networkworld.com/article/3042698/mobile-wireless/global-regulators-shape-the-future-of-lte-u-laa.html

[37] IEEE Project 802, "Coexistence Lessons Learned", *IEEE 802.19-14/0080r2*, 2014.

[38] Ericsson and Qualcomm, "Economic study of the benefits from use of 1452-1492 MHz for a supplemental mobile downlink for enhanced multimedia and broadband services", Tech. Rep., 2011.

[39] Nokia, "The Future of the UHF Band", White Paper, 2016.

[40] http://www.ictregulationtoolkit.org/2.6#s2.6.6

[41] https://arxiv.org/ftp/arxiv/papers/1211/1211.7113.pdf

[42] Accenture, "The Future Communications Service Provider", White Paper, 2015.

[43] ITU, "Digital Dividend: Insights for Spectrum Decisions", White Paper, 2012.

[44] Intel, "Spectrum Sharing: Licensed Shared Access (LSA) and Spectrum Access System (SAS)", White Paper, 2015.

[45] ETSI, "Reconfigurable Radio Systems (RRS); System architecture and high level procedures for operation of Licensed Shared Access (LSA) in the 2300 MHz - 2400 MHz band", ETSI TS 103 235, Tech. Spec., 2015.

[46] http://www.wirelessinnovation.org/fcc-definitions

[47] ETSI, "Reconfigurable Radio Systems (RRS); System requirements for operation of Mobile Broadband Systems in the 2300 MHz - 2400 MHz band under Licensed Shared Access (LSA)", ETSI TS 103 154, Tech. Spec., 2014.

[48] Yrjölä et all, "Evaluation of recent spectrum sharing concepts from business model scalability point of view", *ISBN: 978-1-4799-7452-8*, 2015.

[49] Chaplin&Lehr, "The Path to Market Success for Dynamic Spectrum Access Technology", *ISSN: 0163-6804*, 2007.

[50] ITU-R, "Introduction of new spectrum sharing concepts: LSA and WSD", ITU-R SG 1/WP 1B Workshop, 2014.

[51] Matinmikko et all, "Overview and comparison of recent spectrum sharing approaches in regulation and research", *ISBN: 978-1-4799-2661-9*, 2014.

[52] ETSI, "3GPP TS 32.101 version 13.0.0 Release 13", ETSI TS 132 101, Tech. Spec., 2016.

[53] Nokia, "Overview of LSA activities in ETSI", White Paper, 2014.

[54] Nokia, "Spectrum Developments in Europe", White Paper, 2014.

[55] Coherent, "Report on enhanced LSA, intra-operator spectrum-sharing and micro-area spectrum sharing", Deliverable D4.1, Tech. Rep., 2016.

[56] Lehr, "Spectrum Sharing: getting there from here", CFP Plenary Meeting, 2014.

[57] Google, "Public and Redacted Versions of Request for Confidential Treatment and Complementary Exhibits", 0539-EX-PL-2016, FCC Filing, 2016.

[58] WinnForum, "Understanding the New U.S. 3.5 GHz Band", Webinar, 2015.

[59] NTIA, "3.5 GHz Exclusion Zone Analyses and Methodology", NTIA Report 15-517, Tech. Rep., 2015.

[60] http://www.rcrwireless.com/20150922/opinion/reader-forum-fcc-spectrum-rules-create-business-opportunities-tag10

[61] FCC, "Wireless Telecommunications Bureau and Office of Engineering and Technology Establish Procedure and Deadline for Filing Spectrum Access System (SAS) Administrator(s) and Environmental Sensing Capability (ESC) Operator(s) Applications", DA 15-1426, Public Notice, 2015.

[62] Intel, "Spectrum Sharing For Capacity And Business Growth", Presentation, 2016.

[63] ECC, "Licensed Shared Access (LSA)", ECC Report 205, Tech. Rep., 2014.

[64] http://www.computerworld.com/article/3027925/mobile-wireless/unlicensed-lte-in-the-35ghz-band-could-be-good -for-enterprises.html

[65] METIS II, "Refined scenarios and requirements, consolidated use cases, and qualitative techno-economic feasibility assessment", Tech. Report., 2016.

[66] FCC, "Report And Order And Second Further Notice Of Proposed Rulemaking", FCC 15-47, Tech. Rep., 2015.

[67] WinnForum, "Interim SAS to CBSD Protocol Technical Report-A", WINNF-15-P-0023, Tech. Rep., 2015.

[68] WinnForum, "SAS to CBSD Protocol Technical Report-B", WINNF-15-P-0062, Tech. Rep., 2016.

[69] http://www.macwright.org/2012/10/31/gis-with-python-shapely-fiona.html

[70] http://toblerity.org/shapely/manual.html#

[71] http://toblerity.org/fiona/manual.html

[72] http://www.spatialanalysisonline.com/HTML/

[73] http://www.movable-type.co.uk/scripts/latlong.html

[74] WinnForum, "Interim SAS to SAS Protocol Technical Report-A", WINNF-15-P-0051, Tech. Rep., 2016.

[75] WinnForum, "Interim SAS to SAS Protocol Technical Report-B", WINNF-16-P-0003, Tech. Rep., 2016.

[76] http://www.slideshare.net/pekkaklarck/robot-framework-introduction

[77] https://blog.codecentric.de/en/2016/01/robot-framework-tutorial-2016-installation/

A Appendix 1: SAS-CBSD Simulation Script

```
*** Settings ***
Library           OperatingSystem
Library           Dialogs
Library           Remote    http://${ADDRESS}:${PORT}    WITH NAME    SASServer
Library           Remote    http://${ADDRESS}:${PORT2}    WITH NAME    CBSD1
Variables         Variables/variables.py

*** Variables ***
${ADDRESS}        127.0.0.1
${PORT}           8270
${PORT2}          8272

*** Test Cases ***
0.CBSD1 Initialization
CBSD1.Initialize CBSD    CBSD_a1

1.CBSD1 Unknown->R1->R2->Registered
CBSD1.Send R1 Request
SASServer.Listen And Respond R1
CBSD1.Receive R2 Response

2.CBSD1 Registered->G1->G2->Granted
CBSD1.Send G1 Request
# SASServer.Listen And Respond G1 Excl Zone    ${EXCLUSIONZONE}
SASServer.Listen And Respond G1 ESC    ${USSBunkerHill}
CBSD1.Receive G2 Response

3.CBSD1 Granted->H1->H2->Transmission
CBSD1.Send H1 Request
SASServer.Listen And Respond H1
CBSD1.Receive H2 Response

4 CBSD1 Transmission->L1->L2->Registered
CBSD1.Send L1 Request
SASServer.Listen And Respond L1
```

```
CBSD1.Receive L2 Response

5.CBSD1 Registered->D1->D2->Unknown
CBSD1.Send D1 Request
SASServer.Listen And Respond D1
CBSD1.Receive D2 Response
```

B Appendix 2: SpectrumAccessSystem.py

```
#!/usr/bin/env python

import os
import json
from robot.libraries.BuiltIn import BuiltIn
from robot.libraries.String import String
from SASLocationService import SASLocationService
from SASDatabase import SASDatabase

rbtParamPath = "C:\\SAS\\Param\\"
rbtMsgPath = "C:\\SAS\\Protocol\\"
rbtTempPath = "C:\\SAS\\Transmission\\"
rbtSavedPath = "C:\\SAS\\Saved\\"
rbtSavedFileServer1 = "save_server1.shp"
rbtSavedFileZone1 = "save_zone1.shp"
rbtSavedFileCBSD1 = "save_cbsd1.shp"

class SpectrumAccessSystem(object):
"""
Spectrum Access System
"""
def __init__(self, uid="default"):
"""create SAS server instance with given unique identifier"""
self.uid = uid
self.cbsdId = None

def return_uid(self):
```

```
"""return UID"""
return self.uid

"""
CBSD Configuration----------------------------------------------------------------
"""
def initialize_CBSD(self, cbsdId):
"""initialize CBSD paramters by reading from a file"""
cbsd_obj = SASDatabase().initialize_CBSD_object(cbsdId)
# save cbsd id to self
self.cbsdId = cbsdId
# save CBSD dynamic obj to file
cbsd_obj["fccId"] = cbsdId
SASDatabase().save_CBSD(cbsd_obj, cbsdId)

"""
CBSD Operation--------------------------------------------------------------
"""
def send_r1_request(self):
"""send R1 request by loading msg by given name"""
# load own cbsd data
cbsd_obj = SASDatabase().load_CBSD(self.cbsdId)
# set R1 request data based on own param
request_name = self.cbsdId + "_R1"
r1request = self.load_SAS_message_template("R1_RegistrationRequest")
r1request["registrationRequest"]["fccId"] = cbsd_obj["fccId"]
r1request["registrationRequest"]["cbsdCategory"] = cbsd_obj["cbsdCategory"]
r1request["registrationRequest"]["callSign"] = cbsd_obj["callSign"]
r1request["registrationRequest"]["userId"] = cbsd_obj["userId"]
r1request["registrationRequest"]["airInterface"] = cbsd_obj["airInterface"]
r1request["registrationRequest"]["cbsdManufacturer"] = cbsd_obj["cbsdManufacturer
r1request["registrationRequest"]["cbsdSerialNumber"] = cbsd_obj["cbsdSerialNumber
r1request["registrationRequest"]["maxNumberOfGrant"] = cbsd_obj["maxNumberOfGrant
r1request["registrationRequest"]["sensingCapability"] = cbsd_obj["sensingCapabili
r1request["registrationRequest"]["installationParam"] = cbsd_obj["installationPar
#send R1 request
```

```
self.send_SAS_message(r1request, request_name)

def receive_r2_response(self):
cbsd_obj = SASDatabase().load_CBSD(self.cbsdId)
#listen for R2 msg
expected_suffix = "R2_" + self.cbsdId
r2Name = self.try_search_json_by_suffix(rbtTempPath, expected_suffix)
r2Name = r2Name[:-5]
#Read R2 msg
r2msg = self.receive_SAS_message(r2Name)
err = r2msg["registrationResponse"]["error"]["errorCode"]
#save message content to database
cbsd_obj["error"]["errorCode"] = err
if err == "0":
cbsd_obj["operationState"] = "Registered"
else:
pass
SASDatabase().save_CBSD(cbsd_obj, self.cbsdId)

def send_g1_request_1(self, opParam):
# send G1 request if operation state is Registered
cbsd_obj = SASDatabase().load_CBSD(self.cbsdId)
if cbsd_obj["operationState"] == "Registered":
g1Request = self.load_SAS_message_template("G1_CBSDGrantRequest")
# load operation parameter
cbsdOpParam = self.load_param(opParam)
# setup G1 msg and send
g1Request["grantRequest"]["cbsdId"] = cbsd_obj["fccId"]
g1Request["grantRequest"]["operationParam"] = cbsdOpParam
self.send_SAS_message(g1Request, self.cbsdId + "_G1")
# save OpParam requested to own database
cbsd_obj["operationParam"] = cbsdOpParam
SASDatabase().save_CBSD(cbsd_obj, self.cbsdId)
else:
pass         # do nothing if operation state is not Registered
```

```
def send_g1_request(self):
# send G1 request if operation state is Registered
cbsd_obj = SASDatabase().load_CBSD(self.cbsdId)
if cbsd_obj["operationState"] == "Registered":
g1Request = self.load_SAS_message_template("G1_CBSDGrantRequest")
# setup G1 msg and send
g1Request["grantRequest"]["cbsdId"] = cbsd_obj["fccId"]
g1Request["grantRequest"]["operationParam"] = cbsd_obj["grant"]["operationParam"]
self.send_SAS_message(g1Request, self.cbsdId + "_G1")
else:
pass         # do nothing if operation state is not Registered

def receive_g2_response(self):
cbsd_obj = SASDatabase().load_CBSD(self.cbsdId)
#listen for G2 msg
expected_suffix = "G2_" + self.cbsdId
g2Name = self.try_search_json_by_suffix(rbtTempPath, expected_suffix)
g2Name = g2Name[:-5]
#Read G2 msg
g2msg = self.receive_SAS_message(g2Name)
#save message content to database
cbsd_obj["grant"] = g2msg["grantResponse"]
if cbsd_obj["grant"]["error"]["errorCode"] == "0":
cbsd_obj["operationState"] = "Granted"
else:
pass
SASDatabase().save_CBSD(cbsd_obj, self.cbsdId)

def send_h1_request(self):
cbsd_obj = SASDatabase().load_CBSD(self.cbsdId)
if cbsd_obj["operationState"] == "Granted":
h1Request = self.load_SAS_message_template("H1_HeartbeatRequest")
# setup G1 msg and send
h1Request["heartbeatRequest"]["cbsdId"] = cbsd_obj["fccId"]
h1Request["heartbeatRequest"]["grantId"] = cbsd_obj["grant"]["grantId"]
if cbsd_obj["operationStatusReq"] == True:
```

```
h1Request["heartbeatRequest"]["operationState"] = cbsd_obj["operationState"]
h1Request["heartbeatRequest"]["operationParam"] = cbsd_obj["grant"]["operationPar
else:
pass
self.send_SAS_message(h1Request, self.cbsdId + "_H1")
else:
pass         # do nothing if operation state is not Granted

def receive_h2_response(self):
cbsd_obj = SASDatabase().load_CBSD(self.cbsdId)
#listen for H2 msg
expected_suffix = "H2_" + self.cbsdId
h2Name = self.try_search_json_by_suffix(rbtTempPath, expected_suffix)
h2Name = h2Name[:-5]   #remove .json from file name
#Read H2 msg
h2msg = self.receive_SAS_message(h2Name)
#save message content to database
if h2msg["heartbeatResponse"]["grantId"] == cbsd_obj["grant"]["grantId"]:    #if
cbsd_obj["grant"]["error"]["errorCode"] = h2msg["heartbeatResponse"]["error"]["er
if h2msg["heartbeatResponse"]["error"]["errorCode"] == "0":  #if error status is
cbsd_obj["operationState"] = "Transmission"
else:
pass
else:
pass
SASDatabase().save_CBSD(cbsd_obj, self.cbsdId)

def send_l1_request(self):
# send L1 request if operation state is Transmission or Granted
cbsd_obj = SASDatabase().load_CBSD(self.cbsdId)
if cbsd_obj["operationState"] == "Transmission" or cbsd_obj["operationState"] ==
l1Request = self.load_SAS_message_template("L1_RelinquishRequest")
# setup G1 msg and send
l1Request["relinquishmentRequest"]["cbsdId"] = cbsd_obj["fccId"]
l1Request["relinquishmentRequest"]["grantId"] = cbsd_obj["grant"]["grantId"]
self.send_SAS_message(l1Request, self.cbsdId + "_L1")
```

```
else:
pass         # do nothing if operation state is not Registered

def receive_l2_response(self):
cbsd_obj = SASDatabase().load_CBSD(self.cbsdId)
#listen for L2 msg
expected_suffix = "L2_" + self.cbsdId
l2Name = self.try_search_json_by_suffix(rbtTempPath, expected_suffix)
l2Name = l2Name[:-5]   #remove .json from file name
#Read H2 msg
l2msg = self.receive_SAS_message(l2Name)
#save message content to database
if l2msg["relinquishmentResponse"]["grantId"] == cbsd_obj["grant"]["grantId"]:
cbsd_obj["grant"]["error"]["errorCode"] = l2msg["relinquishmentResponse"]["error"
if l2msg["relinquishmentResponse"]["error"]["errorCode"] == "0":  #if error statu
cbsd_obj["operationState"] = "Registered"
else:
pass
else:
pass
SASDatabase().save_CBSD(cbsd_obj, self.cbsdId)

def send_d1_request(self):
# send L1 request if operation state is Registered
cbsd_obj = SASDatabase().load_CBSD(self.cbsdId)
if cbsd_obj["operationState"] == "Registered":
d1Request = self.load_SAS_message_template("D1_DeregisterRequest")
# setup G1 msg and send
d1Request["deregistrationRequest"]["cbsdId"] = cbsd_obj["fccId"]
self.send_SAS_message(d1Request, self.cbsdId + "_D1")
else:
pass         # do nothing if operation state is not Registered

def receive_d2_response(self):
cbsd_obj = SASDatabase().load_CBSD(self.cbsdId)
#listen for L2 msg
```

```
expected_suffix = "D2_" + self.cbsdId
d2Name = self.try_search_json_by_suffix(rbtTempPath, expected_suffix)
d2Name = d2Name[:-5]   #remove .json from file name
#Read H2 msg
d2msg = self.receive_SAS_message(d2Name)
#save message content to database
if d2msg["deregistrationResponse"]["cbsdId"] == cbsd_obj["fccId"]:    #if grant I
cbsd_obj["error"]["errorCode"] = d2msg["deregistrationResponse"]["error"]["errorC
if d2msg["deregistrationResponse"]["error"]["errorCode"] == "0":  #if error statu
cbsd_obj["operationState"] = "Unknown"
else:
pass
else:
pass
SASDatabase().save_CBSD(cbsd_obj, self.cbsdId)

"""
SAS Server Opearation-----------------------------------------------------
"""
def listen_and_respond_r1(self, err="0"):
#listen for R1 msg
expected_suffix = "_R1"
firstR1 = self.search_json_by_suffix(rbtTempPath, expected_suffix)
firstR1 = firstR1[:-5]
# Read R1 msg
r1msg = self.receive_SAS_message(firstR1)
#load R2 request template
r2response = self.load_SAS_message_template("R2_RegistrationResponse")
#set respond content
r2response["registrationResponse"]["cbsdId"] = r1msg["registrationRequest"]["fccI
#set error code, error default value is 0 = success
r2response["registrationResponse"]["error"]["errorCode"] = err
#send msg
self.send_SAS_message(r2response, "SAS_R2_" + r2response["registrationResponse"][

def listen_and_respond_g1(self, err="0"):
```

```
#listen for G1 msg
expected_suffix = "_G1"
firstG1 = self.search_json_by_suffix(rbtTempPath, expected_suffix)
firstG1 = firstG1[:-5]
# Read G1 msg
g1msg = self.receive_SAS_message(firstG1)
#load G2 request template
g2response = self.load_SAS_message_template("G2_CBSDGrantResponse")
#set respond content
g2response["grantResponse"]["cbsdId"] = g1msg["grantRequest"]["cbsdId"]
#set error code, error default value is 0 = success
g2response["grantResponse"]["error"]["errorCode"] = err
g2response["grantResponse"]["operationParam"] = g1msg["grantRequest"]["operationP
g2response["grantResponse"]["cbsdId"] = g1msg["grantRequest"]["cbsdId"]
grantId = String().generate_random_string(8)    # generate random 8 digit grant I
g2response["grantResponse"]["grantId"] = grantId
#send msg
self.send_SAS_message(g2response, "SAS_G2_" + g2response["grantResponse"]["cbsdId

def listen_and_respond_g1_excl_zone(self, exclzone):
#listen for G1 msg
expected_suffix = "_G1"
firstG1 = self.search_json_by_suffix(rbtTempPath, expected_suffix)
firstG1 = firstG1[:-5]
# Read G1 msg
g1msg = self.receive_SAS_message(firstG1)
#load G2 request template
g2response = self.load_SAS_message_template("G2_CBSDGrantResponse")
#set respond content
g2response["grantResponse"]["cbsdId"] = g1msg["grantRequest"]["cbsdId"]
#load CBSD data from DB
cbsd_obj = SASDatabase().load_CBSD(g1msg["grantRequest"]["cbsdId"])
# geolocation determination
x = float(cbsd_obj["installationParam"]["longitude"])
y = float(cbsd_obj["installationParam"]["latitude"])
deviceloc = x, y
```

```
outexclzone = self.device_in_exclusion_zone(deviceloc, exclzone)
if outexclzone == "0":
#set error code, error default value is 0 = success
g2response["grantResponse"]["error"]["errorCode"] = "0"
else:
g2response["grantResponse"]["error"]["errorCode"] = "400"
# continue constructing response
g2response["grantResponse"]["operationParam"] = g1msg["grantRequest"]["operationP
g2response["grantResponse"]["cbsdId"] = g1msg["grantRequest"]["cbsdId"]
grantId = String().generate_random_string(8)    # generate random 8 digit grant I
g2response["grantResponse"]["grantId"] = grantId
#send msg
self.send_SAS_message(g2response, "SAS_G2_" + g2response["grantResponse"]["cbsdId

def listen_and_respond_g1_esc(self, incumbent):
#listen for G1 msg
expected_suffix = "_G1"
firstG1 = self.search_json_by_suffix(rbtTempPath, expected_suffix)
firstG1 = firstG1[:-5]
# Read G1 msg
g1msg = self.receive_SAS_message(firstG1)
#load G2 request template
g2response = self.load_SAS_message_template("G2_CBSDGrantResponse")
#set respond content
g2response["grantResponse"]["cbsdId"] = g1msg["grantRequest"]["cbsdId"]
#load CBSD data from DB
cbsd_obj = SASDatabase().load_CBSD(g1msg["grantRequest"]["cbsdId"])
#geolocation determination
x = float(cbsd_obj["installationParam"]["longitude"])
y = float(cbsd_obj["installationParam"]["latitude"])
deviceloc = x, y
fa = float(cbsd_obj["grant"]["operationParam"]["operationFrequencyRange"]["lowFre
fb = float(cbsd_obj["grant"]["operationParam"]["operationFrequencyRange"]["highFr
f_avg = float((fa + fb)/2000000) #convert to MHz
pathlosssuff = self.device_pathloss_to_incumbent_sufficient(deviceloc, f_avg, inc
if pathlosssuff == True:
```

```
g2response["grantResponse"]["error"]["errorCode"] = "0"
else:
g2response["grantResponse"]["error"]["errorCode"] = "400"
# continue constructing response
g2response["grantResponse"]["operationParam"] = g1msg["grantRequest"]["operationP
g2response["grantResponse"]["cbsdId"] = g1msg["grantRequest"]["cbsdId"]
grantId = String().generate_random_string(8)    # generate random 8 digit grant I
g2response["grantResponse"]["grantId"] = grantId
#send msg
self.send_SAS_message(g2response, "SAS_G2_" + g2response["grantResponse"]["cbsdId

def listen_and_respond_h1(self, err="0"):
#listen for H1 msg
expected_suffix = "_H1"
firstH1 = self.search_json_by_suffix(rbtTempPath, expected_suffix)
firstH1 = firstH1[:-5]
# Read G1 msg
h1msg = self.receive_SAS_message(firstH1)
#load G2 request template
h2response = self.load_SAS_message_template("H2_HeartbeatResponse")
#set respond content
h2response["heartbeatResponse"]["cbsdId"] = h1msg["heartbeatRequest"]["cbsdId"]
#set error code, error default value is 0 = success
h2response["heartbeatResponse"]["error"]["errorCode"] = err
h2response["heartbeatResponse"]["operationStatusReq"] = True
h2response["heartbeatResponse"]["operationParam"] = h1msg["heartbeatRequest"]["op
h2response["heartbeatResponse"]["grantId"] = h1msg["heartbeatRequest"]["grantId"]
#send msg
self.send_SAS_message(h2response, "SAS_H2_" + h2response["heartbeatResponse"]["cb

def listen_and_respond_l1(self, err="0"):
#listen for L1 msg
expected_suffix = "_L1"
firstL1 = self.search_json_by_suffix(rbtTempPath, expected_suffix)
firstL1 = firstL1[:-5]
# Read G1 msg
```

```
l1msg = self.receive_SAS_message(firstL1)
#load G2 request template
l2response = self.load_SAS_message_template("L2_RelinquishResponse")
#set respond content
l2response["relinquishmentResponse"]["cbsdId"] = l1msg["relinquishmentRequest"]["
l2response["relinquishmentResponse"]["grantId"] = l1msg["relinquishmentRequest"][
#set error code, error default value is 0 = success
l2response["relinquishmentResponse"]["error"]["errorCode"] = err
#send msg
self.send_SAS_message(l2response, "SAS_L2_" + l2response["relinquishmentResponse"

def listen_and_respond_d1(self, err="0"):
#listen for D1 msg
expected_suffix = "_D1"
firstD1 = self.search_json_by_suffix(rbtTempPath, expected_suffix)
firstD1 = firstD1[:-5]
# Read G1 msg
d1msg = self.receive_SAS_message(firstD1)
#load G2 request template
d2response = self.load_SAS_message_template("D2_DeregisterResponse")
#set respond content
d2response["deregistrationResponse"]["cbsdId"] = d1msg["deregistrationRequest"]["
#set error code, error default value is 0 = success
d2response["deregistrationResponse"]["error"]["errorCode"] = err
#send msg
self.send_SAS_message(d2response, "SAS_D2_" + d2response["deregistrationResponse"

"""
SAS Server Service Determination-----------------------------------------------
"""

def device_in_exclusion_zone(self, devicexy, exclusionzone):
"""return True if device is within given exclusion zone"""
exclude = SASLocationService().device_in_zone(devicexy, exclusionzone)
if exclude == False:
return "0"
```

```
else:
return "300"

def device_distance_to_zone(self, devicexy, zone):
"""calculate CBSD distance to exclusion zone"""
distance = SASLocationService().device_distance_to_zone_km(devicexy, zone)
return distance

def device_distance_to_incumbent(self, devicexy, incumbentxy):
"""calculate CBSD distance to incumbent"""
distance = SASLocationService().device_distance_to_incumbent_km(devicexy, incumbe
return distance

def device_pathloss_to_incumbent_sufficient(self, devicexy, devicefreq, incumbent
"""calculate CBSD distance to incumbent"""
distance = SASLocationService().device_distance_to_incumbent_km(devicexy, incumbe
d = float(distance)
f = float(devicefreq)
plSuff = SASLocationService().pathloss_km_mhz(d, f)
if plSuff > 140:
return True
else:
return plSuff

def device_pathloss(self, devicexy, zone, freqInMHz):
"""return True if device is within given exclusion zone"""
distance = SASLocationService().device_distance_to_zone_km(devicexy, zone)
return SASLocationService().pathloss_km_mhz(int(freqInMHz), distance)

def device_outside_exclusion_zone(self, devicexy, exclusionzone):
"""return True if device is outside of given exclusion zone"""
exclude = SASLocationService().device_in_zone(devicexy, exclusionzone)
if exclude == False:
return "300"
else:
return "0"
```

```
"""
Communication-------------------------------------------------------------
"""
def receive_SAS_message(self, filename):
"""read SAS json msg from given path: rbtTempPath"""
fullfilename = rbtTempPath + filename + '.json'
if os.path.exists(fullfilename):
msg = self.load_json_file(fullfilename)
os.remove(fullfilename)
return msg
else:
BuiltIn().log(self, "Message Not Found:" + " " + fullfilename)

def send_SAS_message(self, data, filename):
"""send SAS json msg to given path: rbtTempPath"""
fullfilename = rbtTempPath + filename + '.json'
msg = self.save_json_file(data,fullfilename)

def load_SAS_message_template(self, filename):
"""load SAS json msg template from given path: rbtMsgPath"""
fullfilename = rbtMsgPath + filename + '.json'
msg = self.load_json_file(fullfilename)
return msg

def load_param(self, filename):
"""load parameter file from given path: rbtParamPath"""
fullfilename = rbtParamPath + filename + '.json'
msg = self.load_json_file(fullfilename)
return msg

"""
Json pasing and saving----------------------------------------------------
"""
def load_json_file(self, filename):
"""Fetch json file from given full filename"""
```

```
file = open(filename, 'r')
data = json.load(file)
return data

def save_json_file(self, data, filename):
"""save data as json file to given full filename"""
file = open(filename, 'w')
file.seek(0) #set file position back
json.dump(data, file)

def search_json_by_suffix(self, path, suffix):
"""save data as json file to given full filename"""
for filename in os.listdir(path):
if filename.endswith(suffix + ".json"):
return filename
break

def try_search_json_by_suffix(self, path, suffix):
"""save data as json file to given full filename"""
for filename in os.listdir(path):
if filename.endswith(suffix + ".json"):
return filename
```

C Appendix 3: SASLocationService.py

```
#!/usr/bin/env python

import os
import math
import fiona
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.basemap import Basemap
from math import radians, cos, sin, asin, sqrt
from shapely.geometry import mapping, Polygon, Point, LinearRing
```

```
class SASLocationService(object):
"""SAS Location Sevice"""
def pathloss_km_mhz(self, freqInMHz, disInKm):
"""
calculate flat fading pathloss based on given frequency in MHz and distance in KM
"""
pathloss = 20 * math.log10(freqInMHz) + 20 * math.log10(disInKm) + 32.44
return pathloss

def get_distance_km(self, point1, point2):
"""
get distance between two points in km by applying haversine formula
Lon W/E lat: N/S
# refer to: http://stackoverflow.com/questions/15736995/how-can-i-quickly-estimat
"""
a = float(point1.x)
b = float(point1.y)
c = float(point2.x)
d = float(point2.y)
# convert decimal degrees to radians
lon1, lat1, lon2, lat2 = map(radians, [a, b, c, d])
# haversine formula
dlon = lon2 - lon1
dlat = lat2 - lat1
a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
c = 2 * asin(sqrt(a))
km = 6371 * c
return km

def device_in_zone(self, devicexy, zone):
"""
return true if given point is within given polygon. Inputs are
devicexy: coordinates of device
zone: coordinates of the zone
"""
```

```
poly = Polygon(zone)
ring = LinearRing(zone)
point = Point(devicexy)
#plot map
us_map = Basemap(llcrnrlon=-93,llcrnrlat=37,urcrnrlon=-82,urcrnrlat=49)
us_map.drawcoastlines()
us_map.drawcountries()
plt.show
#plot zone
x, y = ring.xy
plt.plot(x, y)
#plot point
a = float(point.x)
b = float(point.y)
plt.scatter(a, b)
plt.annotate('CBSD', xy = (a, b), xytext = (-20, 20), textcoords = 'offset points
ha = 'right', va = 'bottom', bbox = dict(boxstyle = 'round,pad=0.5', fc = 'blue',
arrowprops = dict(arrowstyle = '-', connectionstyle = 'arc3,rad=0'))
plt.savefig('C:\SAS\Saved\device_in_zone.png')
if point.within(poly):
return True
else:
return False

def device_distance_to_zone(self, devicexy, zone):
"""
return distance between given point and polygon. Inputs are
devicexy: coordinates of device
zone: coordinates of polygon
"""
poly = Polygon(zone)
point = Point(devicexy)
return point.distance(poly)

def device_distance_to_zone_km(self, devicexy, zone):
"""
```

```
return distance between given point and polygon in km. Inputs are
devicexy: coordinates of device
zone: coordinates of polygon
"""
poly = Polygon(zone)
ring = LinearRing(zone)
point = Point(devicexy)
# find closest point from poly to point
pol_ext = LinearRing(poly.exterior.coords)
d = pol_ext.project(point)
p = pol_ext.interpolate(d)
closest_point_coords = list(p.coords)[0]
closest_point = Point(closest_point_coords)
return self.get_distance_km(point, closest_point)

def device_distance_to_incumbent_km(self, devicexy, incumbentxy):
"""
return distance between given cbsd and incumbent. Inputs are
devicexy: coordinates of device
incumbentxy: coordinates of incumbent
"""
incumbent = Point(incumbentxy)
point = Point(devicexy)
#plot map
#us_map = Basemap(llcrnrlon=-119,llcrnrlat=22,urcrnrlon=-64,urcrnrlat=49)
us_map = Basemap(llcrnrlon=-85,llcrnrlat=35,urcrnrlon=-64,urcrnrlat=49)
us_map.drawcoastlines()
us_map.drawcountries()
plt.show
#plot point
a = float(point.x)
b = float(point.y)
plt.scatter(a, b)
plt.annotate('CBSD1', xy = (a, b), xytext = (-20, 20), textcoords = 'offset point
ha = 'right', va = 'bottom', bbox = dict(boxstyle = 'round,pad=0.5', fc = 'blue',
arrowprops = dict(arrowstyle = '-', connectionstyle = 'arc3,rad=0'))
```

```
#plot incumbent
a = float(incumbent.x)
b = float(incumbent.y)
plt.scatter(a, b)
plt.annotate('Aegis', xy = (a, b), xytext = (-20, 20), textcoords = 'offset point
ha = 'right', va = 'top', bbox = dict(boxstyle = 'round,pad=0.5', fc = 'yellow',
arrowprops = dict(arrowstyle = '-', connectionstyle = 'arc3,rad=0'))
plt.savefig('C:\SAS\Saved\device_distance_to_incumbent_km.png')
return self.get_distance_km(point, incumbent)

def save_point_to_shapefile(self, xy, name, file):
"""
Save given point coordinates to file. Inputs are
xy: coordinates of device
name: description of device
file: location of shp file
"""
schema = { 'geometry': 'Point', 'properties': { 'name': 'str' } }
with fiona.open(file, "w", "ESRI Shapefile", schema) as output:
point = Point(xy)
output.write({
'properties': {'name': name},
'geometry': mapping(point)
})

def save_zone_to_shapefile(self, zone, name, file):
"""
Save given point coordinates to file. Inputs are
zone: coordinates of zone (as polygon)
name: description of zone
file: location of shp file
"""
schema = { 'geometry': 'Polygon', 'properties': { 'name': 'str' } }
with fiona.open(file, "w", "ESRI Shapefile", schema) as output:
poly = Polygon(zone)
output.write({
```

```
'properties': {'name': name},
'geometry': mapping(poly)
})
```

D Appendix 4: SASDatabase.py

```
#!/usr/bin/env python

import os
import json

rbtDBMetaPath = "C:\\SAS\\Database\\Meta\\"
rbtDBOperaPath = "C:\\SAS\\Database\\Operation\\"
rbtDBInitPath = "C:\\SAS\\Database\\Init\\"

class SASDatabase(object):
"""SAS Database"""
"""
CBSD related data operations ----------------------------------------------------
"""
def save_CBSD(self, data, cbsdid):
"""send SAS json msg to given path: rbtDBOperaPath"""
fullfilename = rbtDBOperaPath + cbsdid + '.json'
msg = self.save_json_file(data,fullfilename)

def load_CBSD(self, cbsdid):
"""load SAS json msg template from given path: rbtDBOperaPath"""
fullfilename = rbtDBOperaPath + cbsdid + '.json'
""" try to load CBSD """
try:
cbsdObj = self.load_json_file(fullfilename)
except:
cbsdObj = self.create_CBSD_object()
return cbsdObj

def create_CBSD_object(self):
```

```
fullfilename = rbtDBMetaPath + "CBSDMeta" + '.json'
cbsdObj = self.load_json_file(fullfilename)
return cbsdObj

def initialize_CBSD_object(self, cbsdId):
fullfilename = rbtDBInitPath + "CBSDInit_" + cbsdId + '.json'
cbsdObj = self.load_json_file(fullfilename)
return cbsdObj

"""
Grant related data operations ------------------------------------------------
"""
def create_grant_object(self):
fullfilename = rbtDBMetaPath + "GrantMeta" + '.json'
grant = self.load_json_file(fullfilename)
return grant

"""
Json pasing and saving-------------------------------------------------------
"""
def load_json_file(self, filename):
"""Fetch json file from given full filename"""
file = open(filename, 'r')
data = json.load(file)
file.seek(0) #set file position back
return data

def save_json_file(self, data, filename):
"""save data as json file to given full filename"""
file = open(filename, 'w+')
file.write(json.dumps(data))
file.close()
```